

Faculty of Science and Engineering

Generic High Level Feature Detection Techniques Using
Multi-Modal Spatial Data

Richard Leslie Palmer

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

December 2015

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made. This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signed: 

Date: 16 December 2015

Generic High Level Feature Detection Techniques Using Multi-Modal Spatial Data

by

Richard Leslie Palmer

This thesis is presented for the Degree of
Doctor of Philosophy of Curtin University

December, 2015

Abstract

Two dimensional colour and greyscale images have classically been the prevalent form of data used in object and pattern recognition research. Newer methods of data collection such as mobile-mapping, which uses vehicle mounted laser scanners and stereophotogrammetric camera arrays, produce 2-D images combined with 3-D spatial data resulting in datasets consisting of colourised point clouds and depth maps. The added modality of spatial depth provided by these datasets is motivating renewed interest in approaches to feature extraction and object recognition, particularly where explicitly acquired depth information can be used to mitigate issues related to object scale.

The detection, recognition and localisation of objects of interest from very large image and point cloud datasets is a common task across many different industries. Local Government agencies, utility companies, commercial organisations and private parties maintain asset registers detailing their particular distributed infrastructures (*e.g.* traffic lights, telegraph poles, garbage bins). The costs associated with the collection and maintenance of these datasets are significant. Improvements in the automated detection and localisation of objects using such datasets are therefore of keen interest to these users, as these techniques have the potential to reduce the administrative burden associated with the upkeep of their asset registers.

Many computer vision systems have been developed to assist with object recognition, but most of these systems rely only upon 2-D images of scenes and are unable to make use of the extra modality of depth acquired by mobile-mapping systems. This thesis reports on two object and pattern recognition techniques that were initially developed to address the task of object detection and localisation in 2-D images, but have been enhanced herein to take advantage of the explicit availability of depth data so as to improve their accuracy in object classification / detection.

In the first of these methods, a popular and well performing feature extraction method (the *Histograms of Oriented Gradients*) is redeveloped to undertake the efficient extraction of features from arbitrarily located image subregions. These descriptors are compared in an object classification framework in a scale invariant manner without the need to rescale image

data. In the second method, object detection is carried out using a parts-based detection technique based on *Implicit Shape Models* and *Hough Forests*. The technique is modified to incorporate depth data in three different ways so as to enhance object detection accuracy. Importantly, the modifications do not require adjustments to the underlying algorithm.

A final chapter introduces a new implementation of the *Euclidean Distance Transform* algorithm that supercedes existing state-of-the-art approaches in its computational complexity and runtime performance. Generalisations of this algorithm are intrinsic to parts-based methods of object detection as well as several other techniques in computer vision and pattern recognition.

All of the techniques described in the thesis are empirically evaluated against newly acquired 3-D datasets generated from mobile-mapping systems, as well as standard 2-D image datasets where appropriate.

Acknowledgements

In no particular order...

I am grateful to my supervisor Professor Geoff West, for having provided me with the opportunity to undertake my doctoral studies in this area. I am also grateful for his and my associate supervisor Tele Tan's time, guidance, support, and entirely unjustified patience with me over the course of my studies (and in particular, the generation of this thesis).

I am grateful to my parents, Jeanette and Kenneth Palmer who have unfailingly supported and invested themselves emotionally and financially in my goals. To the rest of my family and friends, your good natured patience with my overly complicated rants about the most esoteric of abstractions has allowed me to stay focussed and productive. Thank you. Also, if you ever read this, I'll be impressed if you get further than these few paragraphs. Go on. You can do it...

I also want to thank everyone I've met from all of the organisations that have supported me in various ways during the course of my studies. I'm awful with names, but that's why business cards were invented, right?

To Misty. You are my rock and my beacon, and you have shown the most patience of anyone. I hope that when your time comes to write up your thesis, you're better able to keep it together than I was! xxx

Mum. I'm sorry you weren't able to hang around to see me finish this. You dedicated yourself to science to try to make the world a better place. You've set a great example. I dedicate this thesis to you.

Copyright Acknowledgements

Imagery downloaded from the Internet has been used in this thesis in accordance with Section 40 of the Australian Copyright Act (1968) which provides for the fair dealing of artistic work for research or study. The respective copyright holders are hereby acknowledged:

- **NASA's Jet Propulsion Laboratory (Caltech)** photojournal.jpl.nasa.gov/targetFamily/Mars
- **earthmine Australia (Landgate, Western Australia)** www0.landgate.wa.gov.au/maps-and-imagery/imagery/earthmine
- **AAM** www.aamgroup.com

Publications

This thesis includes the following works that have been published during the course of the PhD study:

- Palmer, R., Borck, M., West, G., and Tan, T. (2012). Intensity and Range Image Based Features for Object Detection in Mobile Mapping Data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 315–320.
- Palmer, R., West, G. A. W., and Tan, T. (2012). Scale Proportionate Histograms of Oriented Gradients for Object Detection in Co-Registered Visual and Range Data. In *Proceedings of the International Conference on Digital Image Computing Techniques and Applications (DICTA)*. IEEE.
- Palmer, R., West, G., and Tan, T. (2013). Using depth to extend randomised hough forests for object detection and localisation. In *Digital Image Computing: Techniques and Applications (DICTA), 2013 International Conference on*, pages 1–8.
- Borck, M., Palmer, R., West, G., and Tan, T. (2014). Using depth maps to find interesting regions. In *Region 10 Symposium, 2014 IEEE*, pages 62–67.
- Lin, T., Palmer, R., Xia, J., and McMeekin, D. (2014). Automatic generation of station catchment areas: A comparison of euclidean distance transform algorithm and location-allocation methods. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2014 11th International Conference on*, pages 963–967.

Contents

1	Introduction	1
1.1	Generic Object Recognition	1
1.1.1	Object Classification and Detection	3
1.1.2	The Challenges of Object Recognition	4
1.2	Multi-Modal Data	6
1.3	Contributions and Thesis Outline	7
2	Background	9
2.1	Representing Features	9
2.1.1	Feature Vector Length and Generality	12
2.1.2	Resizing Image Extracts	13
2.1.3	Scale Space and Image Pyramids	14
2.1.4	Using Depth Data for Object Recognition	16
2.2	Machine Learning	17
2.2.1	The Automated Recognition of Patterns	18
2.2.2	Unsupervised versus Supervised Learning	19
2.2.3	Generative versus Discriminative Classification	20
2.2.4	Naïve Bayes	21
2.2.5	Logistic Regression	22
2.2.6	Classification Accuracy: Recall and Precision	25
2.2.7	Combining Classifiers	29
2.2.8	Classifiers Used In This Thesis	30
2.2.9	Support Vector Machines	31
2.3	Object Modelling	33
2.3.1	Modelling Objects by Parts	35
2.3.2	Implicit Shape Models	37
3	Datasets	45
3.1	Dataset Fusion	45
3.2	The Earthmine Dataset	47

3.3	The AAM Dataset	51
3.4	Ground Truthing	55
3.4.1	Application and Workflow Design	55
3.4.2	Task Details	57
3.4.3	Task Evaluation	59
3.5	Evaluating Detection Accuracy	61
3.5.1	Evaluating Object Classification Accuracy	62
3.5.2	Evaluating Object Detection Accuracy	65
3.5.3	Problems with Alternative Evaluation Criteria	66
4	Scale Proportionate Histograms of Oriented Gradients	69
4.1	Pro-HOG Design	70
4.1.1	Histograms of Oriented Gradients	70
4.1.2	The Pro-HOG Algorithm	72
4.1.3	Phase One: Pixel Resolution Contrast Gradient Integral Images	74
4.1.4	Phase Two: Cell Histograms and Final Descriptor Generation	76
4.1.5	Expected Benefits	80
4.2	Testing Pro-HOG’s Parameters	82
4.2.1	General Methodology	86
4.2.2	Gradient Sign Sensitivity	88
4.2.3	Histogram Length	92
4.2.4	Extract Scaling	98
4.2.5	Cell Grid Dimensions	100
4.3	Evaluation Against Many Object Types	106
4.3.1	Experimental Aims	106
4.3.2	Methodological Parameters	108
4.3.3	Object Types	109
4.3.4	Pascal VOC “Car”	110
4.3.5	Pascal VOC “Person”	112
4.3.6	AAM “Car”	114
4.3.7	AAM “Traffic Light”	118
4.3.8	AAM “Triangular Road Sign”	121
4.3.9	AAM “Truck / Van”	124
4.3.10	AAM “Road Light”	126
4.3.11	AAM “Telegraph Pole”	131
4.3.12	AAM “Rectangular Road Sign”	134
4.3.13	AAM “Generic Road Sign”	137
4.3.14	Earthmine “Car”	141
4.3.15	Earthmine “Garbage Bin”	145

4.3.16	Earthmine “Traffic Light”	148
4.3.17	Earthmine “Parking Sign”	150
4.3.18	Earthmine “Traffic Cone”	154
4.4	Observations and Recommendations	160
4.4.1	Emulating HOG	160
4.4.2	Effects of Gradient Sign Sensitivity (GSS)	161
4.4.3	Effects of Scale Invariance (SI)	161
4.4.4	Better Accuracy with Shorter Descriptors	163
4.5	Conclusion	172
4.5.1	Depth Based Features	173
4.5.2	Limitations	175
5	Extending Randomised Hough Forests Using Depth Data	176
5.1	Class Specific Hough Forests	178
5.1.1	Training (Tree Generation)	180
5.1.2	Detection (Vote Aggregation)	183
5.2	Depth Based Extensions	185
5.2.1	Patch / Offset Scaling	185
5.2.2	Depth Surface Features	191
5.2.3	Depth Weighting	193
5.3	Experimental Method	193
5.3.1	Estimating Detected Object Bounds	197
5.3.2	Earthmine Depth Image Preprocessing	198
5.4	Results and Analysis	201
5.4.1	Earthmine “Car”	201
5.4.2	Earthmine “Garbage Bin”	203
5.4.3	Earthmine “Traffic Light”	205
5.4.4	Earthmine “Parking Sign”	207
5.4.5	Earthmine “Traffic Cone”	208
5.4.6	AAM “Car”	210
5.4.7	AAM “Traffic Light”	212
5.4.8	AAM “Triangular Road Sign”	213
5.4.9	AAM “Truck / Van”	215
5.4.10	AAM “Road Light”	216
5.4.11	AAM “Telegraph Pole”	217
5.4.12	AAM “Rectangular Road Sign”	218
5.4.13	Overall Analysis	221
5.5	Conclusion	222
5.5.1	Limitations	223

5.5.2	Further Research	225
6	An Efficient Euclidean Distance Transform	226
6.1	Distance Transform Metrics	228
6.1.1	Difficulties in computing the Euclidean Distance Transform	230
6.1.2	Euclidean Distance Transform Algorithms	232
6.2	The Extended EDT Algorithm	236
6.2.1	First Stage Calculation	236
6.2.2	Second Stage Calculation	237
6.2.3	Algorithm Extensions	242
6.2.4	Unimplemented Modifications	246
6.3	Experimental Methodology	251
6.3.1	Datasets	253
6.3.2	Exactness	257
6.4	Results	260
6.5	Conclusion	270
7	Conclusions	271
7.1	Limitations	273
A	Glossary of Terms	289

List of Figures

1-1	A Mobile Mapping platform: The Earthmine Mars stereophotogrammetric image capture system. Copyright Earthmine Australia 2015.	6
2-1	A simplified representation of the population distributions of two object classes Blue and Red within a two dimensional feature space, and the training data (circles and crosses) used to train a classifier represented by the decision boundary (thick black line) estimating the learned demarcation between the classes. The estimated boundary is an imperfect fit to the true boundary and this will invariably lead to errors in classification.	27
2-2	The positive (blue) class and the negative (red) class population distribution within a two dimensional feature space is shown. Both the linear and the non-linear classifier fit the data, but due to the more accurate fit to the true population boundary given by the non-linear classifier, it offers greater precision and recall accuracy.	32
2-3	Extracting “part” descriptors as part of the codebook generation process for two ISMs encoding “jagged explosion” and “rounded rectangle” object types.	38
2-4	The codebook of “parts” in table 2.1 can be used to detect objects having similar parts to those in the codebook.	39
2-5	Detected wheel “parts” encoded by an Implicit Shape Model cast votes for the location of a car’s centre.	40
2-6	Due to the car’s rotation, the part offset vectors to the object centroid (which are derived from examples without the same degree of rotation) centroid no longer agree and so the centre of the car is less reliably detected by the wheel “parts”.	41
2-7	A partitioning of feature space by a decision tree that approximately matches the regions populated by three different object types (the blue areas).	43
3-1	The Earthmine Mars Collection System collects high resolution stereo panoramic imagery. Copyright Earthmine Australia 2015.	48
3-2	Visualising an example point cloud taken from the Earthmine dataset within the custom point-cloud viewer application.	49

3-3	An example scene taken from the Earthmine dataset showing the quality of the raw image data.	50
3-4	An example of incorrect depth interpolation in the Earthmine dataset.	51
3-5	An example point cloud in the AAM dataset.	52
3-6	An example of scene from the AAM dataset (looking left from the mapping vehicle).	53
3-7	Problems with the AAM dataset including incorrect colour mapping and “stippling” of objects near to the scanning vehicle caused by the inability to register multiple laser scans at close range.	54
3-8	The user interface for the custom ground-truthing application.	57
3-9	Participant work rates and object density	61
3-10	An example of precision-recall curves (extracted from section 4.3.9)	64
3-11	A dense sampling of image subregions is classified. Areas of the image are used to calculate the four core statistics according to how the detection subregions (red boxes) overlap the ground truth bounding boxes (blue boxes).	66
4-1	Conversion of an image matrix (left) to an integral image matrix (right). The integral image allows for sums over arbitrary regions of the image (dark blue region) to be calculated by indexing into the integral image only at the four corners of the subregion.	72
4-2	The main steps of the Pro-HOG feature extraction algorithm.	73
4-3	Image scaling and cell grid definition for HOG (left column) versus Pro-HOG (right column). Image from the PASCAL VOC 2007 “Car” dataset.	77
4-4	Lena Pro-HOG visualisation	80
4-5	A zoomed in image of Lena showing the detail of the eyes in the visualised Pro-HOG image, in particular the contrast magnitude peaks (darker parts) around the eyes and the verticality of the nose bridge given by the strong vertical orientations shown by the histograms in that area.	81
4-6	Sample extracts from the “Person” Pascal VOC 2007 dataset. Extracts have varying dimensions and are resized for display here.	83
4-7	Example images from the “INRIA” dataset (Dalal and Triggs, 2005)	83
4-8	Sample extracts from the “Car” Pascal VOC 2007 dataset. Extracts have varying dimensions and are resized for display here.	84
4-9	Smallest and largest examples from the “Car” Pascal VOC 2007 dataset.	84
4-10	Smallest and largest examples from the “Person” Pascal VOC 2007 dataset.	85
4-11	Sample output from a single round of cross-validation	87

4-12	Contrast gradient binning inside a single 4×4 pixel cell. The histogram bins into which gradient magnitudes are binned differ depending on whether Pro-HOG is sensitive to the direction of the gradient (the lower histogram) or not (the upper histogram).	89
4-13	Pro-HOG precision-recall curves comparing classification accuracy with gradient sign sensitivity enabled and disabled (Linear SVMs).	90
4-14	Pro-HOG precision-recall curves comparing classification accuracy with gradient sign sensitivity enabled and disabled (Linear SVMs) (<i>Original images, plus horizontally reflected images</i>).	91
4-15	Pro-HOG classification precision and recall versus cell histogram length for HOG, and GSS enabled and disabled versions of Pro-HOG. (“Person” Pascal VOC 2007 dataset with linear SVMs).	93
4-16	Pro-HOG classification precision and recall versus cell histogram length for HOG, and GSS enabled and disabled versions of Pro-HOG (“Car” Pascal VOC 2007 dataset with linear SVMs).	94
4-17	Pro-HOG classification precision and recall versus cell histogram length for HOG, and GSS enabled and disabled versions of Pro-HOG. (“Person” Pascal VOC 2007 dataset with non-linear SVMs).	95
4-18	Pro-HOG classification precision and recall versus cell histogram length for HOG, and GSS enabled and disabled versions of Pro-HOG. (“Car” Pascal VOC 2007 dataset with non-linear SVMs).	96
4-19	Recall & Precision against Extract Scale using Pro-HOG and Linear Classifiers	99
4-20	Recall versus cell grid dimensions for the “Person” Pascal VOC 2007 dataset	101
4-21	Precision versus cell grid dimensions for the “Person” Pascal VOC 2007 dataset	103
4-22	Recall versus cell grid dimensions for the “Car” Pascal VOC 2007 dataset	104
4-23	Precision versus cell grid dimensions for the “Car” Pascal VOC 2007 dataset	104
4-24	Distribution of example sizes for the Pascal VOC “Car” object type.	110
4-25	Precision versus Recall using feature descriptors extracted from intensity images of the Pascal VOC 2007 “Car” object type.	111
4-26	Distribution of example sizes for the Pascal VOC “Person” object type.	112
4-27	Precision versus Recall using feature descriptors extracted from intensity images of the Pascal VOC 2007 “Person” object type.	113
4-28	Sample extracts from the AAM “Car” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	114
4-29	Distribution of example sizes for the AAM “Car” object type.	115
4-30	Precision versus Recall using descriptors extracted from intensity images of the AAM “Car” object type.	116
4-31	Precision versus Recall using descriptors extracted from depth images of the AAM “Car” object type.	117

4-32	Sample extracts from the AAM “Traffic Light” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	118
4-33	Distribution of example sizes for the AAM “Traffic Light” object type.	118
4-34	Precision versus Recall using descriptors extracted from intensity images of the AAM “Traffic Light” object type.	119
4-35	Precision versus Recall using descriptors extracted from depth images of the AAM “Traffic Light” object type.	120
4-36	Sample extracts from the AAM “Triangular Road Sign” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	121
4-37	Distribution of example sizes for the AAM “Triangular Road Sign” object type.	121
4-38	Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Triangular Road Sign” object type.	122
4-39	Precision versus Recall using feature descriptors extracted from depth images of the AAM “Triangular Road Sign” object type.	123
4-40	Sample extracts from the AAM “Truck / Van” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	125
4-41	Distribution of example sizes for the AAM “Truck / Van” object type.	126
4-42	Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Truck / Van” object type.	126
4-43	Precision versus Recall using feature descriptors extracted from depth images of the AAM “Truck / Van” object type.	127
4-44	Sample extracts from the AAM “Road Light” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	128
4-45	Distribution of example sizes for the AAM “Road Light” object type.	128
4-46	Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Road Light” object type.	129
4-47	Precision versus Recall using feature descriptors extracted from depth images of the AAM “Road Light” object type.	130
4-48	Sample extracts from the AAM “Telegraph Pole” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	131
4-49	Distribution of example sizes for the AAM “Telegraph Pole” object type.	132
4-50	Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Telegraph Pole” object type.	133
4-51	Precision versus Recall using feature descriptors extracted from depth images of the AAM “Telegraph Pole” object type.	134
4-52	Sample extracts from the AAM “Rectangular Road Sign” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	135
4-53	Distribution of example sizes for the AAM “Rectangular Road Sign” object type.	135

4-54	Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Rectangular Road Sign” object type.	136
4-55	Precision versus Recall using feature descriptors extracted from depth images of the AAM “Rectangular Road Sign” object type.	137
4-56	Sample extracts from the AAM “Generic Road Sign” dataset. Images are resized to fixed width for display here, but have varying actual dimensions. . .	138
4-57	Distribution of example sizes for the AAM “Generic Road Sign” object type.	138
4-58	Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Generic Road Sign” object type.	139
4-59	Precision versus Recall using feature descriptors extracted from depth images of the AAM “Generic Road Sign” object type.	140
4-60	Sample extracts from the Earthmine “Car” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	141
4-61	Distribution of example sizes for the Earthmine “Car” object type.	142
4-62	Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Car” object type.	143
4-63	Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Car” object type.	144
4-64	Sample extracts from the Earthmine “Garbage Bin” dataset. Images are resized to fixed width for display here, but have varying actual dimensions. . .	145
4-65	Distribution of example sizes for the Earthmine “Garbage Bin” object type. .	145
4-66	Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Garbage Bin” object type.	146
4-67	Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Garbage Bin” object type.	147
4-68	Sample extracts from the Earthmine “Traffic Light” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	148
4-69	Distribution of example sizes for the Earthmine “Traffic Light” object type. .	148
4-70	Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Traffic Light” object type.	149
4-71	Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Traffic Light” object type.	151
4-72	Sample extracts from the Earthmine “Parking Sign” dataset. Images are resized to fixed width for display here, but have varying actual dimensions. . .	152
4-73	Distribution of example sizes for the Earthmine “Parking Sign” object type. .	152
4-74	Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Parking Sign” object type.	153
4-75	Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Parking Sign” object type.	154

4-76	Sample extracts from the Earthmine “Traffic Cone” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.	155
4-77	Distribution of example sizes for the Earthmine “Traffic Cone” object type. .	155
4-78	Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Traffic Cone” object type.	156
4-79	Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Traffic Cone” object type with extraction cell-block dimensions of 4×4	157
4-80	Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Traffic Cone” object type.	158
4-81	Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Traffic Cone” object type with extraction cell-block dimensions of 4×4	159
4-82	Percentage improvement in accuracy (F1 score) of Pro-HOG (Scale Independent) from Pro-HOG (Resizing) over all object classes at cell-block dimensions of 8×8	162
4-83	Percentage improvement in precision of the Pro-HOG (Scale Independent) and Pro-HOG (Max Resized) extractors from the Pro-HOG (Median Resized) extractor for object types from the AAM dataset (at object specific optimal cell-block dimensions).	168
4-84	Percentage improvement in recall of the Pro-HOG (Scale Independent) and Pro-HOG (Max Resized) extractors from the Pro-HOG (Median Resized) extractor for object types from the AAM dataset (at object specific optimal cell-block dimensions).	169
4-85	Percentage improvement in precision of the Pro-HOG (Scale Independent) and Pro-HOG (Max Resized) extractors from the Pro-HOG (Median Resized) extractor for object types from the Earthmine dataset (at object specific optimal cell-block dimensions).	170
4-86	Percentage improvement in recall of the Pro-HOG (Scale Independent) and Pro-HOG (Max Resized) extractors from the Pro-HOG (Median Resized) extractor for object types from the Earthmine dataset (at object specific optimal cell-block dimensions).	171
5-1	Fixed magnitude offsets cause votes to be cast inaccurately when matching object parts with different scales.	190
5-2	The pixel dimensions of scaled offsets are found according to the dimensions of the detected parts. This causes votes to be cast closer to the true locations of the object reference points.	191
5-3	Missing and erroneous depth information in the Earthmine dataset.	199

5-4	After preprocessing the depth image in figure 5-3(b) using the average dimensions of the Earthmine “Traffic Light” object type.	200
5-5	Precision versus Recall for Earthmine “Car” detection using CSHF with and without depth extensions.	201
5-6	A representative query image from the Earthmine “Car” dataset, and generated Hough vote maps for the four different experimental configurations.	202
5-7	Precision versus Recall for Earthmine “Garbage Bin” detection using CSHF with and without depth extensions.	204
5-8	A representative query image from the Earthmine “Garbage Bin” dataset, and generated Hough vote maps for the four different experimental configurations.	205
5-9	Precision versus Recall for Earthmine “Traffic Light” detection using CSHF with and without depth extensions.	206
5-10	A representative query image from the Earthmine “Traffic Light” dataset, and generated Hough vote maps for the four different experimental configurations.	206
5-11	Precision versus Recall for Earthmine “Parking Sign” detection using CSHF with and without depth extensions.	207
5-12	A representative query image from the Earthmine “Parking Sign” dataset, and generated Hough vote maps for the four different experimental configurations.	208
5-13	Precision versus Recall for Earthmine “Traffic Cone” detection using CSHF with and without depth extensions.	209
5-14	A representative query image from the Earthmine “Traffic Cone” dataset, and generated Hough vote maps for the four different experimental configurations.	209
5-15	Precision versus Recall for AAM “Car” detection using CSHF with and without depth extensions.	210
5-16	A representative query image from the AAM “Car” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.	211
5-17	Precision versus Recall for AAM “Traffic Light” detection using CSHF with and without depth extensions.	212
5-18	A representative query image from the AAM “Traffic Light” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.	213
5-19	Precision versus Recall for AAM “Triangular Road Sign” detection using CSHF with and without depth extensions.	214
5-20	A representative query image from the AAM “Triangular Road Sign” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.	214
5-21	Precision versus Recall for AAM “Truck / Van” detection using CSHF with and without depth extensions.	215

5-22	A representative query image from the AAM “Truck / Van” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.	216
5-23	Precision versus Recall for AAM “Road Light” detection using CSHF with and without depth extensions.	217
5-24	A representative query image from the AAM “Road Light” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.	218
5-25	Precision versus Recall for AAM “Telegraph Pole” detection using CSHF with and without depth extensions.	219
5-26	A representative query image from the AAM “Telegraph Pole” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.	219
5-27	Precision versus Recall for AAM “Rectangular Road Sign” detection using CSHF with and without depth extensions.	220
5-28	A representative query image from the AAM “Rectangular Road Sign” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.	220
6-1	Chessboard, city block (Manhattan), and Euclidean distance metrics.	229
6-2	Applying distance transforms using different metrics to an image with a single, centred feature pixel. The city block (a), chessboard (b) and Chamfer 3-4 (c) metrics produce results suffering from anisotropic effects. Only the Euclidean distance metric (d) results in an isotropic (rotationally invariant) distance map. (Images are contrast stretched for clarity.)	231
6-3	Discretisation of Euclidean distances leads to problems of non-locality.	232
6-4	Calculation of the location of intersection along the row being equidistant from foreground points \mathbf{p} and \mathbf{q} by parabola intersection (Felzenszwalb and Huttenlocher, 2004)	238
6-5	Direct calculation of the location of intersection along the row being equidistant from foreground points \mathbf{p} and \mathbf{q} by perpendicular bisection (Wang and Tan, 2013)	239
6-6	Process of comparing the intersection of the Voronoi region at row R for new point \mathbf{q} against previously stored points \mathbf{g} and \mathbf{p} (iteration along R at i in each diagram moving left to right)	241
6-7	When the first foreground point at l is parsed, previously stored candidate points at locations h, i, j and k are discarded.	243

6-8	A newly parsed point \mathbf{p} is discarded early if its location of intersection t along row y is either beyond the effective end of the array, or the orthogonal distance from row y to a foreground point already in column t is $\leq \mathbf{p} - \langle t, y \rangle $	244
6-9	In 6-9(a), contiguous regions of equal row value can be ignored to avoid unnecessary updates (only endpoint locations need be stored). In 6-9(b), the presence of a point lower than the contiguous row values complicates the calculation of the intersection location.	247
6-10	Sample test images from the Fabbri <i>et al.</i> (2008) dataset	255
6-11	Sample images from the Mars dataset courtesy of NASA/JPL-Caltech (2014)	256
6-12	Sample test images from the Natural (Misc) dataset	257
6-13	Sample test images from the Random Lines dataset	258
6-14	Inexact calculation by an implementation of Maurer <i>et al.</i> (2003)'s distance transform algorithm Fabbri <i>et al.</i> (2008) (see the top right corner of figure 6-14(c))	259
6-15	Inexact calculation of the EDT by Maurer <i>et al.</i> (2003)'s algorithm (Fabbri <i>et al.</i> , 2008) on the Mars dataset. Original image courtesy of NASA/JPL-Caltech (2014).	263
6-16	Box-plots showing per algorithm median and interquartile range results for the Fabbri <i>et al.</i> (2008) dataset	265
6-17	Box-plots showing per algorithm median and interquartile range results for the MARS dataset	266
6-18	Box-plots showing per algorithm median and interquartile range results for the MARS EDGE dataset	267
6-19	Box-plots showing per algorithm median and interquartile range results for the NATURAL dataset	268
6-20	Box-plots showing per algorithm median and interquartile range results for the RANDOM LINES dataset	269

List of Tables

2.1	An example codebook of parts from two different object types (Θ and Φ) based on the extracted parts of the “training” examples in figure 2-3.	39
3.1	Object types and counts from the ground-truthing exercise on the AAM dataset.	58
3.2	Participant 1 work rates	60
3.3	Participant 2 work rates	60
4.1	Object Classes used for Object Classification Evaluation	109
4.2	Classification Results – Pascal VOC “Car” (Intensity descriptors)	111
4.3	Classification Results – Pascal VOC “Person” (Intensity descriptors)	112
4.4	Classification Results – AAM “Car” (Intensity descriptors)	115
4.5	Classification Results – AAM “Car” (Depth descriptors)	116
4.6	Classification Results – AAM “Traffic Light” (Intensity descriptors)	119
4.7	Classification Results – AAM “Traffic Light” (Depth descriptors)	120
4.8	Classification Results – AAM “Triangular Road Sign” (Intensity descriptors)	122
4.9	Classification Results – AAM “Triangular Road Sign” (Depth descriptors) . .	123
4.10	Classification Results – AAM “Truck / Van” (Intensity descriptors)	124
4.11	Classification Results – AAM “Truck / Van” (Depth descriptors)	127
4.12	Classification Results – AAM “Road Light” (Intensity descriptors)	129
4.13	Classification Results – AAM “Road Light” (Depth descriptors)	130
4.14	Classification Results – AAM “Telegraph Pole” (Intensity descriptors)	132
4.15	Classification Results – AAM “Telegraph Pole” (Depth descriptors)	134
4.16	Classification Results – AAM “Rectangular Road Sign” (Intensity descriptors)	136
4.17	Classification Results – AAM “Rectangular Road Sign” (Depth descriptors) .	137
4.18	Classification Results – AAM “Generic Road Sign” (Intensity descriptors) . .	139
4.19	Classification Results – AAM “Generic Road Sign” (Depth descriptors) . . .	140
4.20	Classification Results – Earthmine “Car” (Intensity descriptors)	142
4.21	Classification Results – Earthmine “Car” (Depth descriptors)	143
4.22	Classification Results – Earthmine “Garbage Bin” (Intensity descriptors) . .	146
4.23	Classification Results – Earthmine “Garbage Bin” (Depth descriptors)	147
4.24	Classification Results – Earthmine “Traffic Light” (Intensity descriptors) . . .	149

4.25	Classification Results – Earthmine “Traffic Light” (Depth descriptors)	150
4.26	Classification Results – Earthmine “Parking Sign” (Intensity descriptors) . .	151
4.27	Classification Results – Earthmine “Parking Sign” (Depth descriptors)	153
4.28	Classification Results – Earthmine “Traffic Cone” (Intensity descriptors) . . .	155
4.29	Classification Results – Earthmine “Traffic Cone” (Depth descriptors)	158
4.30	Pascal VOC 2007 dataset: comparing Pro-HOG’s scale independence versus image resizing using optimal cell-block dimensions.	165
4.31	AAM dataset: comparing Pro-HOG’s scale independence versus image resizing using optimal cell-block dimensions.	167
4.32	Earthmine dataset: comparing Pro-HOG’s scale independence versus image resizing using optimal cell-block dimensions.	170
5.1	Object Classes used for Object Detection Evaluation	194
5.2	The four experimental configurations for object detection on each object type.	195
6.1	Distance Transform Algorithms Evaluated	253
6.2	Datasets used for evaluation of Euclidean Distance Transform algorithms . .	254
6.3	Fabbri <i>et al.</i> (2008) dataset results	260
6.4	MARS dataset results	260
6.5	MARS EDGE dataset results	261
6.6	NATURAL dataset results	261
6.7	RANDOM LINES dataset results	261

List of Algorithms

1	Pro-HOG's calculation of oriented contrast magnitudes over the vertical and horizontal difference maps of an input image (assumes single channel for simplicity).	75
2	EDT Stage 1: Fast in-place calculation of 1-D squared Euclidean distances	237
3	Calculation of the location of row intersection using integer division	245
4	EDT Stage 2: Per row identification of candidate nearest points	248
5	EDT Stage 2a: Skip over initial points at left	249
6	EDT Stage 2b: Update the stack of candidate nearest points with new point q	249
7	EDT Stage 3: Squared Euclidean distance calculation over non-foreground	250

Chapter 1

Introduction

Evolution by natural selection has, over millions of years, developed mechanisms of visual recognition that enable animals to recognise a diverse range of patterns, shapes, and objects in their environments. Human beings (and other primates) can recognise objects in 2-D images, even when the objects are presented in unfamiliar environments or orientations (Logothetis *et al.*, 1994; Bühlhoff and Edelman, 1992). Computer vision research into *Generic Object Recognition* rests upon the assumption that visual recognition can, to some degree, be synthesised within the existing computational paradigm. That is, it is assumed that recognition can be comprised of essentially serial tasks whereby input images are parsed to generate, for example, lists of segmented scene elements with associated object labels.

It is not known if computers are sufficiently technologically advanced at this time to be able to solve the generic object recognition problem. However, advances have been made concerning recognition tasks specific to particular object instances or classes of objects that express variation within constraints, such that a model of the type can be generated and stored to compare against input data.

1.1 Generic Object Recognition

Historically, the problem of generic object recognition has been viewed as attempting to accurately recognise and classify previously unknown objects from 2-D images where no prior knowledge of the scene is available (Lowe, 1987). Knowledge of the scene includes the nature of the scene itself, as well as the kind, number, and distribution of objects of interest in the scene. The problem is compounded if the data are degraded due to, for example, non-optimal lighting conditions, the presence of “clutter” (irrelevant features), or awkward viewing

angles that can result in the salient image features becoming occluded. The generic object recognition system is meant to recognise which features of the input data comprise objects that are of interest to the user. One of the most challenging issues is how the system can know that particular features are of interest to the user without being explicitly told. If the user is interested in generic objects, then the system must have within it a codification of “objectness” so that the elements of an input scene that express this characteristic can be identified.

Computer vision theorists and neuroscientists have tried to understand how the human visual system operates as a way of trying to understand how objects are characterised in images. The modern age of object recognition research began when neuroscientist David Marr developed a more exacting approach to computer vision and object recognition systems research (Marr, 1982). This framework, known as Marr’s paradigm, allowed researchers to focus their attention on particular aspects of the problem (Barrow *et al.*, 1981). The most popular idea about how human vision works is that two different modes of attention in the brain are combined; bottom-up signals determine the salient or interesting regions in a scene, while top-down task focussed mechanisms direct attention to known objects (Bar *et al.*, 2006; Mechelli *et al.*, 2004; Navalpakkam and Itti, 2006; Itti and Koch, 2001; Oliva *et al.*, 2003; Borenstein and Ullman, 2008).

Modern object recognition systems are developed along similar lines with the important top-down requirement that the recognition system has some *a priori* knowledge of the nature of the objects it is supposed to recognise *i.e.* those objects that are of interest to the user. In analogy to the human engineering of flight in comparison to the understanding of the evolved mechanics of flight in the animal world, the development of practically useful automated vision systems does not have to be restrained by our currently limited understanding of how the human visual system is so adroit in the task of generic object recognition. The most fundamental way in which generic object recognition is constrained is in the replacement of the concept of generic objects with descriptions of the kinds of features the system should recognise in the input data. This transforms the generic object recognition problem into a task focussed search for specific features or objects of interest. More practically, this means that the accuracy of object recognition systems can be evaluated by comparing their outputs with human labellings of objects in the same input data. As long as the system can provide comparable accuracy to a human in its ability to recognise the objects of interest, how the system achieves this (whether or not it has any similarities to the way in which visual processing is undertaken in the brain) is irrelevant.

There are several application areas that benefit greatly from top-down task driven approaches to object recognition. The examples below in one way or another search for specific features,

patterns, or objects in their input data:

- Automated surveillance (Aggarwal and Ryoo, 2011),
- Robot navigation (Kruger *et al.*, 2007),
- Autonomous driving (Geiger *et al.*, 2012),
- Content-Based Image Retrieval (CBIR) (Datta *et al.*, 2008),
- Image-Based Rendering (IBR) (Snavely *et al.*, 2010),
- Photographic manipulation (Chen *et al.*, 2009), and,
- Augmented Reality (AR) (Palmese and Trucco, 2008)

In academic and commercial institutions the world over, task oriented automatic object recognition systems are now a burgeoning area of research and development.

1.1.1 Object Classification and Detection

The top-down, task focussed approach to recognising objects of interest in input data is more commonly described as *object classification*. Relative to generic object recognition, this is the simpler task of assigning labels from a database of object archetypes to one or more input images. In this task, an image extract is presented to a *recogniser* or *classifier*. The classifier has access to a database of generic models of different object archetypes, where each model specifies the characteristic information about an archetype in sufficient detail to be able to distinguish an instance of that type in an input image. The database may also contain an entry for the *None* or *Unknown* type, which is assigned to an image extract when the characteristics of the image evaluated by the classifier cannot be appropriately assigned to any of the available archetypes defined by the stored models. In theory, this allows every input image extract to be associated with the most appropriate label indicating the type of object.

Object detection develops the classification task further to search for one or more specific instances or examples of a particular object type in an image or a set of images. If the aim is to detect an example of a particular object type, a model of that type is stored which

represents the expected variation of the type’s exemplars. In object *detection*, it is unknown *a priori* if an object is present in the image and, if it is present, where in the image it is located. Further, it may not be necessary to classify the object’s type, merely to understand if *some* object from a subset of types is present.

Object classification is often combined with detection because the detection of an object at some location in an image nearly always requires comparing some extracted features from that part of the image with a stored model of the object. In this thesis, the term “object detection” is always used to refer to the detection of an example of an object’s archetype (or simply type or class) rather than a particularly specified object.

From this point forward, the term “object recognition” is used to refer to either object classification or object detection or their combination. Object recognition in the generic sense will no longer be discussed.

1.1.2 The Challenges of Object Recognition

Taking a top-down task driven approach to object recognition is the first step in making any recognition task computationally tractable. However, there remain significant challenges that must be overcome depending on the specific nature of the recognition task.

Even if the objects of interest can be enumerated, it is not immediately clear how the models of these object types should be encoded for the recognition task without first understanding the nature of the input data that will be parsed by the recognition system. The models must be able to provide positive responses to the variety of 2-D projections and scales that the real 3-D objects are shown in, since the models must be compared to the input data (which nominally consist of 2-D image projections). Models that can only describe objects in particular orientations, or at particular scales are not useful if the input imagery never displays the objects in those orientations or at those scales. Similarly, if the objects of interest are composed of articulated parts (*e.g.* people), or if there is wide variation in the appearance of the object type (in terms of morphology, texture, or colour), then those object configurations will remain unrecognised (*i.e.* false negatives) by models that do not adequately describe the possible variations. The difficulty of recognition also increases as the variety of object types to be detected increases, since this entails a larger model database resulting in lengthier searches.

Objects are frequently occluded by other objects or features in the input imagery and back-

ground clutter can confuse the recognition process, resulting in the erroneous detection of objects where none exist (*i.e.* false positives). The input imagery will also likely contain features that cannot be identified by the recognition system as belonging to modelled object types or known scene elements and so the identification of all objects in a scene is generally not possible.

Finally, the process of capturing the input data is itself imperfect (*e.g.* specular highlights or shadowing resulting from non-ideal lighting), meaning that the recognition system must be robust to a wide range of problematic imaging conditions.

Many of these issues can be entirely negated by constraining the scope of the recognition task to be undertaken. Consider the case of an automated quality assurance system that aims to detect potential problems with some known manufactured rigid part, copies of which pass by on a conveyor belt where upon an image of the part is captured and passed to the system for processing. The specified recognition task involves identifying that the part is present, that it appears in a certain orientation, and that it is sufficiently similar in appearance to the model of the part stored by the system to pass the quality assurance parameters.

In this scenario, a database of object type models is not needed because only a single object is being modelled. If the object is not detected according to the single stored model, this should indicate a potential problem to be investigated. Since the frequency of objects passing by on the conveyor belt is known and there should be no other objects present on the conveyor belt, it is not necessary to account for the possibility of occlusions or clutter. Anything that seems on initial processing as though it is not an image of the object kept on file, should result in the system raising a problem to be investigated. The location of the image capture system with respect to the conveyor belt, and thus the objects on the conveyor belt, is known, and so the expected orientation of the objects is known within tight parameters. Moreover, the part is non-articulated and so should have consistent morphology. The range of expected colours and textures of the object can also be specified exactly. Further, the objects will always be artificially lit to the same degree of brightness and from the same angles. This simplifies the modelling of the projected appearance of the object since any variation outside of these strict parameters should cause the system to fail to recognise the object, and this may be suggestive of a fault.

The strict task constraints around this particular recognition problem mean that developing an automated solution is eminently feasible. The recognition task has in effect been recast as a much more simple specific object detection and template matching problem. This example demonstrates that it is often sufficient to develop techniques to solve only particular aspects of the object recognition problem given the constraints of the recognition task and knowledge



Figure 1-1: A Mobile Mapping platform: The Earthmine Mars stereophotogrammetric image capture system. Copyright Earthmine Australia 2015.

of the problem context (*i.e.* the nature of the objects to be detected and the input image data).

1.2 Multi-Modal Data

Most recent object recognition research has been concerned with the task of detecting and classifying objects in 2-D colour (RGB) and grey-scale (intensity) images of scenes. Imaging techniques such as stereophotogrammetry and laser scanning are beginning to see wider use, especially in mobile mapping which is the process of generating geospatial data using vehicles fitted with LiDAR, laser, radar, or photogrammetric remote sensing arrays. Mobile mapping systems generate dense 3-D point clouds of scanned environments. Depending on the sensor configuration used, these point data can be combined with coregistered colour information. Figure 1-1 shows the Earthmine Mars mobile mapping stereophotogrammetric image capture system mounted on a vehicle.

Much research is being conducted into investigating how 3-D geometric object models can assist in object recognition tasks. However, the increasing availability of depth as an extra modality in these image capture and mobile mapping systems means that there is great potential for the combination of depth and 2-D image data to be used as the input for many algorithms from fields related to object recognition such as pattern recognition, machine learning, and computer vision. In particular, depth data as an extra modality can help to constrain object recognition problems, increasing the accuracy of existing recognition systems, or even helping to make certain kinds of recognition feasible. This thesis focuses on how depth

data can assist in these tasks.

1.3 Contributions and Thesis Outline

Object recognition is a large and complex field of research encompassing a wide variety of fundamental techniques from several areas of computer science. This thesis documents research that was undertaken to develop some specific techniques in these areas as they relate to the task of detecting examples from small sets of object types in scene data captured by mobile imaging and mapping systems. The data are comprised of colour images with pixel coregistered depth measurements.

The thesis is broken down as follows: Chapter 2 provides some necessary definitions and details concerning certain essential elementary techniques and concepts that are referenced throughout the thesis. For convenience, a glossary of terms is provided in appendix A.

Chapter 3 details the nature of the datasets used in this research together with the efforts undertaken to generate the necessary training data required to conduct the experiments detailed in later chapters. This chapter also describes the criteria and metrics used to evaluate the accuracy of the object classification and detection experiments undertaken in this thesis.

Chapter 4 describes the development of a new feature extraction algorithm designed to be of particular benefit in object recognition tasks where depth data are explicitly available. The chapter compares this new method in the context of object classification against an existing popular feature extraction method.

In chapter 5, the presence of individual object parts are detected using a randomised decision tree based algorithm that combines aspects of classification and regression to discover the locations of the comprising objects of interest. The algorithm is extended to take advantage of explicit depth information so as to more accurately localise the detected objects of interest.

Chapter 6 investigates the *Euclidean Distance Transform* algorithm. While not directly used in other areas of this thesis, the algorithm is of fundamental importance in many areas of computer vision and image processing more generally. In this chapter, the existing state-of-the-art algorithm is analysed and modifications are introduced that improve upon its algorithmic efficiency. In a thorough empirical evaluation of the technique against a wide variety of data

and alternative algorithms, the adjustments are shown to have practical benefits in improving processing efficiency on real world datasets. The algorithm is also theoretically scalable to spatial data having greater than two dimensions.

Chapter 7 summarises the major findings of this thesis.

Chapter 2

Background

This chapter provides descriptions of essential terms and concepts that are used throughout this thesis. The background material is divided into three areas of intrinsic importance in automated object and pattern recognition. Section 2.1 describes the concept of features, how they are generated from image data, and how they are used for object and pattern recognition. Section 2.2 describes the underlying principles and mathematics involved in machine learning, the kind of machine learning algorithms used in this thesis, and some of the challenges and pitfalls involved in using these techniques. Section 2.3 describes how the concepts detailed in the previous two sections can be integrated to successfully model whole object categories, and how these models are used to detect and locate objects in query images. Appendix A contains a glossary of terms.

2.1 Representing Features

In general, an object recognition scheme can be broken up into two main facets, both of which work in concert to assist in the accurate detection of the objects of interest. These facets are the methods by which objects or patterns of interest are encoded, and the methods by which they are compared. Both of these aspects depend upon the fundamental concept of *feature descriptors* or *feature vectors*. This section describes the nature of features, how they are represented, and how they are used in object / pattern recognition.

The term *descriptor*, or *region descriptor* is used especially when feature vectors are extracted from image subregions having a certain specific character of interest, or from whole objects or specific object parts. The algorithms that output feature vectors are called *feature extractors*. In this thesis, feature extractors are developed to encode both the global char-

acteristics of whole objects (chapter 4), and the characteristics of localised parts of objects (chapter 5).

When performing generic object recognition, finding a match between an object class (some stored model of an object that the user wishes to detect) and some input data requires evaluating measures of similarity that are neither too strict, nor too accommodating. If the similarity measures are too strict, then object instances that are considered viable examples of the class will go undetected (false negatives). If the similarity measures are too accommodating of variability in object representation, then too much input data will be mistakenly identified as valid examples of the object type (false positives).

Features are characteristics of an image that can be extracted at either a *global* level over the whole of an object or image, or at a *local* level specified by subregions or particular points in an image. The simplest example of a feature is the value of a pixel that represents its colour or intensity (in the case of a grey scale image). A feature vector (or descriptor) can then be defined as the concatenation of the individual values of a group of pixels such that $\mathbf{v} = \langle v_1, v_2, v_3, \dots \rangle$. If such a feature vector is created from all of the pixel values bounded by some rectangular region of an image that contains an object of interest, then the feature vector can be thought of as a descriptor for that object. If it is then required that a new image \mathbf{I} be queried to detect the presence of that object, the detection can be carried out by constructing feature descriptors $\mathbf{u}_i \in \mathbf{I}$ from all similarly sized bounding rectangles in \mathbf{I} and optimising a similarity measure $f(\mathbf{v}, \mathbf{u}_i)$. The similarity measure is generally defined as some arithmetic comparison of the corresponding entries in the two feature vectors. For example, if \mathbf{v} and \mathbf{u}_i are normalised, the inner product of the two vectors is often used because the definition of this operation performs a component wise comparison of the vector values. For the inner product, f is maximised when $\mathbf{v} = \mathbf{u}_i$.

In this naïve example, the type of features being used (the pixel values themselves) do not afford much variation in the representation of different object instances as they appear in the image data. Natural variation in the appearance of the objects that ought not to influence the accuracy of the detection scheme can undermine the ability of the matching scheme to identify objects of a similar type. For example, differences in brightness or contrast in the query images will change the values in the generated feature vectors \mathbf{u}_i meaning that \mathbf{u}_i cannot equate to \mathbf{v} and the similarity measure f will give less optimal values. Features must be used that are more robust (*i.e.* features that capture aspects of objects that are relevant to the detection of those objects, while ignoring those aspects that are irrelevant to encoding the presence of the object). Object detectors that can accommodate greater degrees of variability are desired; not just in lighting, but also in scale, orientation, and viewpoint, as well as differences in the morphology, colour, or texture of the objects themselves. For the

purposes of detection, features should seek to encode only the *salient* aspects of an object’s presence, not the incidental aspects. As such, the pixel values themselves are often not used, and measures such as local contrast are often preferred since measures of difference between neighbouring pixel values tend to be less affected by issues of lighting (in particular), than the values themselves.

In general, since features can be derived from any characteristics of an image, any measured feature value exists as a point in a range of possible values along that feature dimension. If multiple different feature types are derived from a single part of the image, the concatenation of those feature values into a descriptor can be viewed as a multidimensional measurement in the *feature space*, defined by the range of values that can be taken on by those features.

The choice of the length of the feature vectors is partly determined by the quality of the data. Combinations of multiple different feature types have shown success (Opelt *et al.*, 2006; Kittler *et al.*, 1998; Gehler and Nowozin, 2009). Individual features are likely to be present in many places in the data and appear in many different object types, so reliance upon a single feature is unlikely to offer sufficient discriminatory power between object classes. Using many different types of feature along with their spatial relationships can enhance the discriminatory power of the descriptors in object classification (Campbell and Flynn, 2001). Multiple different features increases the likelihood of being able to linearly separate object categories, potentially improving object classification performance (this is discussed further in section 2.2). Even though intuitively there does not appear to be much advantage to generating descriptors that combine large numbers of features in different combinations, there is evidence showing that over-complete samplings of certain kinds of features from image data can result in improved recognition performance (Lecun *et al.*, 1998; Riesenhuber and Poggio, 1999; Lowe, 2004; Dalal and Triggs, 2005; Krizhevsky *et al.*, 2012). However, the run-time costs of comparing large numbers of features may become prohibitive since the complexity of object search increases exponentially for each new feature added to the descriptors.

Descriptors that encode a very large number of features will suffer from the *curse of dimensionality*. This is the problem of being able to accurately infer an estimation of the population distribution of an object class over the whole of some high dimensional feature space. To achieve this, an object training set with cardinality that is exponential in the dimensionality of the descriptors is required. Such problems can be mitigated by using hierarchical methods of feature comparison however (Nister *et al.*, 2006; Wu *et al.*, 2008).

The quality of an object’s representation in an image affects the ability of a feature extractor to accurately capture the salient information about that object. Features that are more robust to such problems can be extracted in two ways. Firstly, features can be selected that

are naturally less susceptible to the type of variation expected in the image data. This means having sufficient knowledge of the nature and context of the data to understand how prevalent the different forms of representational variability are.

Secondly, features can be selected for extraction only from certain areas of the image, or from relatively larger regions. For example, it may be the case that that homogeneous subregions of an image (*e.g.* in colour or texture) do not in general provide much salient information about the presence of the objects of interest. In this case, extracting features from those areas and involving them in the matching process may only serve to inject spurious information into the object matching scheme. Alternatively, depending on the recognition task, such homogeneous areas may be very indicative of the presence of a certain type of object. *Interest Point* (or *Keypoint*) detectors are often used to help identify the parts of an image that might be of more use. However, it is necessary to understand the nature of the data and the objects to be detected so that an interest point detector can be employed that is appropriate for the recognition task.

2.1.1 Feature Vector Length and Generality

Feature vectors are compared element wise to one another and this necessitates that they be of equal length. For local features, the individual entries must also spatially correspond for a meaningful comparison. Feature vector length must therefore be determined prior to use, which means that the feature extraction algorithm must produce vectors of fixed length from arbitrarily located image subregions. Objects may also be present at differing scales and orientations, and so it is often necessary to extract features from image subregions that are not fixed in their dimensions. For global feature vectors, individual feature values are calculated over the whole of the subregion (an example feature might be the ratio of “dark” to “light” pixels over the whole of the subregion) and so it matters less that different image subregions contain different numbers of pixels, or that the subregions have different aspect ratios.

A digital image of an object is already a discrete sampling of the true data; the value of each pixel is an aggregate of the photons collected on the corresponding region of the camera’s image sensor. Extracting feature values that are derived from digital image subregions that are larger than a single pixel can be construed as another form of sampling. Depending on the feature extraction method, a particular feature value will be informed by several pixels in a discrete local area. It is often the case that local features are collected from subregions of the projected region of an object in an image. If these subregions are relatively small

compared to the object region, the generated features will be more invariant because groups of fewer pixels cannot encode as much variation as larger groups. These smaller patches can be less susceptible to some of the object representation challenges stated previously. Increasing the proportion of the object that is needed to derive any given feature value reduces the detectable resolution of the object using those same features. That is, the object must already be apparent in the image at a minimum pixel resolution to supply enough information to the feature extraction algorithm to derive feature values that allow for sufficiently accurate discrimination of the object from other scene elements.

Objects at higher pixel resolutions present more detail. This might help to more accurately model the feature value's distribution for a given type of object (if enough high resolution examples are provided). Alternatively, higher resolution examples may include spurious aspects of the objects that vary too much between instances (expressing too much intraclass variation), resulting in feature distributions that poorly represent the object of interest at that resolution. This can result in too many false positive detections when trying to match instances of the same object type. Objects at lower resolutions may not include enough characteristic information to accurately model the distribution of the feature values because not enough of the salient structure of the object is visible. Determining a good object resolution for the purposes of feature extraction and detection is particularly problematic where the scale of an object is not fixed.

If a feature extraction mechanism is used that fixes the sampling resolution of the features (*i.e.* the local region used to derive that feature has statically defined pixel dimensions), constructing feature descriptors of fixed length can be achieved by altering the dimensions of the input image. However, resizing the image data prior to feature extraction can introduce other problems.

2.1.2 Resizing Image Extracts

Resizing an image (or an image subregion) can change the quality of information, affecting the ability of the feature extractor to accurately encode salient characteristics about the features or objects in the image. Dollar *et al.* (2014) showed that the information content of images that have been resized (as sampled by a contrast gradient extractor) changes in proportion to the degree of scaling. Although scaling an image up (scaling by a factor ≥ 1) allows for the extraction of features with information content that is linear in the resizing factor (that is, information is not lost), down scaling an image (scaling by a positive factor < 1) tends to result in the loss of structural information and an attendant loss of information in the extracted

descriptors. The degree of information loss in a downsampled image is larger than implied by the downsampling factor. For contrast gradient magnitude normalised features, Dollar *et al.* (2014) showed that the degree of loss is even more severe, which is a problem since such normalisation can result in improved object detection accuracy. This suggests that feature extractors that resize images in an initial preprocessing step risk corrupting the information content of the extracted features, especially where resizing acts mainly to downsample images from their originally sampled pixel resolution. Up sampling images may also degrade the accuracy of feature encoding because of image interpolation methods, which (in general) act over fixed size image regions irrespective of scale. This may cause local contrast differences to become “smoothed” and less distinctive. The significance of this effect over larger aggregated regions is not well understood.

Another problem not directly addressed by Dollar *et al.* (2014) has to do with the accuracy of feature encoding between images that are ostensibly similar in nature but have different native resolutions and are being resized to the same fixed pixel dimensions by different scaling factors. The resizing operations will affect the content of the two images differently particularly if one of the images must be scaled up and the other down. If scaling an image up, the result will appear blurred because a standard interpolation technique must be used by the rescaling algorithm to infer the content between neighbouring pixels in the original image, which are moved further apart in the rescaled image. Using a simple image scaling algorithm, it is generally not possible for the information content of the image to remain proportional to the number of pixels in the image as the number of pixels used to represent the image changes. An image that is scaled down will be represented by a reduced number of pixels and the information content about neighbouring pixels must be aggregated, losing detail. It is hypothesised that scaling images from the same class of object by different scaling factors may introduce artefacts in the images that diminish the accuracy of the features being extracted when used in the context of object recognition. This hypothesis is investigated in detail in chapter 4 through the development of a feature extraction algorithm that does not require images to be resized prior to the generation of a descriptor.

2.1.3 Scale Space and Image Pyramids

Without explicit knowledge about the distance to points of interest in an image of a scene, managing appearance variation due to scale is typically performed by undertaking detection over a predetermined range of image sizes termed the *scale-space* (Witkin, 1984). The differently sized images are stored within a dedicated data structure termed an *image pyramid*. The search for objects of interest in scale-space is then conducted by parsing the smallest

up to the largest of the images in the pyramid. If positive responses are seen at the smaller scales, the larger scale images can be investigated to assist in the verification of these detections. This means that the object models (or at least parts of the models) must also be comparable to the imagery at different scales. This is a popular approach when using parts-based object models. The more detailed parts of the model describing the object type are encoded at higher resolutions while a “root” part that describes the whole appearance of the object is encoded at a lower resolution. Object hypotheses are initially generated in lower resolution images using the root part of the model, which assists to efficiently threshold the image into more and less promising regions. The more detailed higher resolution object part encodings can then be used to validate these hypotheses. Felzenszwalb *et al.* (2008) used this approach with object models having deformable parts. The detection of the higher resolution parts in configurations acceptable to the model parameters validated the initial detections by minimising the deformation of the independently detected object parts using a *generalised distance transform* – see chapter 6.

To improve detection accuracy, more levels in the image pyramid are needed so search can be undertaken more thoroughly. This is expensive in memory and processing efficiency however, and so a trade off is necessary to achieve the desired levels of accuracy within processing constraints. Image pyramids that represent scale-space with too few levels means that objects in the query images having scales that are not represented in the image pyramid, may go undetected.

The resizing of individual images or image extracts to fit larger or smaller pixel areas as required for the extraction of descriptors is not especially computationally intensive until the operation has to be performed many thousands of times. This can be the case in object detection when the feature extractor must parse a very large number of separate subregions of a query image. If explicit depth information is not available, then image pyramids must be used to account for the possible presence of objects at different scales, and detection must be carried out over the differently sized images, multiplying the number of resize operations carried out over subregions of the image. Very fast implementations of scale-space image pyramids exist for use in object recognition (Dollar *et al.*, 2014; Crowley *et al.*, 2002; Eaton *et al.*, 2006), but avoiding their use entirely by taking advantage of direct depth information will always be faster since it avoids redundant reprocessing of the image at different scales.

Feature extraction techniques can be incorporated into interest point detectors that look for the presence of certain image characteristics at different scales, selecting the “strongest” of these at some scale level as points of interest in an image. Such scale invariant techniques include the Scale Invariant Feature Transform (SIFT) (Lowe, 1999, 2004), the Gradient Location Orientation Histogram (GLOH) (Mikolajczyk and Schmid, 2005), Contrast Context

Histograms (CCH) (Huang *et al.*, 2008), and Speeded-Up Robust Features (SURF) (Bay *et al.*, 2008). These interest point detectors locate regions of the query image, such as corners or edges, that can be used as seed regions for the extraction of descriptors centred on and around these points. Corners and edges are identified by large changes in image intensity over localised areas of typically two or three pixels in diameter. These interest point detectors are effective because corners and edges encode structural information (as well as colour and texture) about a scene, and contrast is relatively stable under changing lighting conditions compared to absolute brightness, which is very susceptible to changes in illumination.

The Harris detector (Harris and Stephens, 1988) and the FAST (Features from Accelerated Segment Test) detector (Rosten and Drummond, 2006) are two widely used interest point detectors that aim to identify strong corner-like changes in intensity in the input image. However, unlike the interest point detectors that incorporate scale invariance in their design, the MSER, Harris and FAST detectors respond well only to gradient changes at predetermined pixel dimensions, and so scale invariance cannot be achieved without processing an image at different sizes. Not all interest point detectors designed without scale variance in mind are redundant for interest point detection. The Maximally Stable Extremal Region (MSER) interest point detector by Matas *et al.* (2004) is unlike the Harris and FAST detectors in that it aims to discover blob-like structures rather than corner-like points in the input image through the gradual variation of an intensity value threshold parameter t . Regions that express minimal change over a large enough range of t are deemed to be stable regions of potential interest for further processing. This method of detecting interesting points in the image data is intrinsically robust to scale variance as it does not detect features such as corners at a fixed pixel resolution.

2.1.4 Using Depth Data for Object Recognition

Multi-modal approaches to object classification and detection are relatively new because the technology to produce range data that is accurately co-registered with 2-D colour images is still maturing. Range (or depth) data give distances from the camera plane to objects in a scene. A depth map represents this dimension of the data as a 2-D image. In this thesis, points in the depth maps have values specified in real world units (typically metres). For visualisation, the depth maps are contrast scaled to more easily indicate changes in distance between scene elements.

Several feature extraction methods make explicit use of depth data such as Normal Aligned Radial Features (NARF) (Steder *et al.*, 2010), Tripod operators (Pipitone and Adams, 1993),

Splash features (Stein and Medioni, 1992) (and other features that encode information about object surface normals (Li and Guskov, 2007)). Depth data have also been used to assist in guiding the object detection scheme over the image. By combining intensity based features with range, images can be segmented into planar regions in the Z axis. This allows appearance based features be scaled in accordance with the detected depth, and guides the detection process resulting in improved detection accuracy (Rapus *et al.*, 2008; Wei *et al.*, 2011). Other approaches to multi-modal object detection distinguish between features that are extracted from 2-D images and features that are derived from depth, which are then combined into a probabilistic framework (Gould *et al.*, 2008).

Incorporating direct depth measurements into the feature extraction process, features can be extracted that are more robust to changes in scale. Range information can be used to make inferences about the viewpoint or the orientation of an object, and thus the RGB features that are likely to be present in the corresponding image. Spin images (Johnson and Hebert, 1999) identify a 3-D reference point on an object to determine how a 2-D histogram should be rotated for feature comparison.

The use of global image context (termed *gist*) has been used in 2-D colour images to inform the likely presence of local features of interest (Murphy *et al.*, 2006). Using range information as contextual background information for an object has also been used with some limited success (Wang *et al.*, 2012). He *et al.* (2008) showed that contrast information can be encoded over multiple scales prior to the extraction of contrast gradient features. Although this can help in making the extracted features more robust against changes in scale, it does not help to discern the actual size of objects.

A large number of different feature extraction techniques have been developed, often to leverage particular aspects of the data that are available for a specific recognition task. Several disciplined comparisons between the relative effectiveness of different feature extraction techniques in the context of object classification have been undertaken (Tuytelaars and Mikolajczyk, 2007; Mikolajczyk and Schmid, 2005; Li and Allinson, 2008; Yamazaki and Fels, 2009; Knopp *et al.*, 2010), but it is difficult to make general inferences about the relative efficacy of different feature extractors when employed outside of their specific recognition tasks.

2.2 Machine Learning

This section provides a brief overview of *machine learning* and *pattern recognition* in the context of object recognition. The aim of this section is not to provide an overarching summary

of these (expansive) areas of research, but to introduce the general concepts and the specific techniques and terms from these domains that are used in this thesis.

2.2.1 The Automated Recognition of Patterns

Object recognition sits within the broader field of pattern recognition since objects are described as patterns of data represented by lists of scalar values (feature vectors). Mathematically, the recognition of a pattern X means it is determined as being an instance of some Y . That is, a function $h : X \mapsto Y$ encodes the relationship between the X input values and the Y output values. The function h is called the hypothesis and it conducts *classification* if Y is discrete and *regression* if Y is continuous. X can also be discrete or continuous. In the remainder of this thesis, X is continuous and the function h is termed the classification function since Y is usually discrete (although in chapter 5, regression is also undertaken alongside classification). The problem of automated pattern and object recognition is in large part the problem of how to define the function h .

The classical approach to generating the classification function h is to manually define lists of rules or heuristics that can act on the input variable X to output the correct value of Y . These rules stipulate conditions for the expected variations in the input variable X . The inflexible nature of these conditions means that these “expert systems” are successful in only limited domains where the training data that have been used to empirically derive the rules for the classification function have a close similarity with the test data. Further, the lists of rules are usually defined by a human expert who has constrained domain knowledge and the resulting classification function will reflect the expert’s subjective biases. It is also difficult to incorporate new observations into such systems, especially where such data contradict existing rules in the classifier. It is not necessarily clear how and where changes should be made in the system to account for adjustments in knowledge based on new observations. Such systems are also not well suited to the task of regression because decisions made by expert systems are usually categorical in nature.

The more modern approach to generating the classification function h is to do so implicitly through techniques that automatically learn models of data. These automatically generated models are then compared against new observations of X and the models that best fit the observations are reported back to the user. All of these machine learning techniques are statistical in nature in that they estimate their decision functions based upon large samples of data. As such, machine learning algorithms are based upon a probabilistic framework, which offers greater nuance in the decisions generated by classification functions learned in

this way. It is often possible to formulate the classification function to output a mapping to Y associated with some real valued confidence or likelihood of the classification being correct. Thresholding on this value can be used to make the classifier more or less sensitive to its inputs (this is exploited in the generation of precision-recall graphs described in section 3.5).

Unlike the classical approach, the automated process of learning the classification function also lends itself more readily to the inclusion of new observations in its models. Finally, automated learning is advantageous when the data are complex and the correlations and relationships in the data are not obvious. The field of machine learning is concerned with the development of algorithms that are able to generate successful classification functions of the form $h_\theta : X \mapsto Y$, or equivalently, to approximate a classification in the form of a probability $P(Y | X; \theta)$.

There are two aspects to the classification function h_θ . The framework for the model encoded by the function h is typically chosen according to expert knowledge. This model represents the function's mathematical or algorithmic specification. θ , which is usually a vector quantity, represents the parameters to the classifier. One kind of model h can be parameterised for many different kinds of classification task. The model parameters θ are learned from data. The manner in which these parameters are learned may be *supervised* if the data are associated with Y values / labels, or *unsupervised* if no such Y value associations are given. When there is no risk of confusion, the classification function is specified without θ such as $h(\mathbf{x})$ (with h taking a vector valued input).

2.2.2 Unsupervised versus Supervised Learning

Classifiers compare the similarity of an observation encoded by a feature vector to labelled pattern archetypes (models) encoded in the classifier. These models are usually produced offline during a *training* or *learning* phase. There are two main types of learning: *supervised*, and *unsupervised*.

Unsupervised learning methods are often used where no labelled observations are available. That is, for some set of N dimensional observations (typically provided as vectors), there are no associated labels that categorise the observations. Unsupervised methods can be used to analyse these data and perform automated grouping (or *clustering*) of the observations. There are two main problems with unsupervised learning approaches. The first is in how two data points should be compared for similarity. Typically, Euclidean distances are used but this assumption can be complicated if the measurements are not standardised, causing some

scalar components of the vectors to hold greater influence and biasing clustering along certain dimensions where the scalar components are less variable. It is also unclear how linearly separable the vectors are. Many of the individual feature values may be correlated, again causing a clustering bias. As a result, the first step in this process is often to undertake some form of dimensionality reduction such as Principal Components Analysis (PCA) to understand the main dimensions of variance in the data. This can also help to simplify the classifier and increase its accuracy by reducing the number of dimensions in which clustering need be performed (the relative density of the data in the modelled dimensions is increased).

The second problem with unsupervised learning is that the number of clusters the data should be segregated into is unknown *a priori*. Even if a visualisation of the data is available (*e.g.* if the data can be visualised in two or three dimensions through dimensionality reduction), two (or more) clusters of data points along a particular dimension do not necessarily mean that those features were measured from two (or more) different cohorts. It might simply be that the feature value x_i has a bi-modal (or multi-modal) distribution given the group from which they were measured. That is, $P(X = x_i | Y = y_k)$ does not necessarily have a single mode. Often, there is a temptation to assume that features are normally distributed when no such evidence exists.

In supervised learning, classifier training is undertaken by supplying a large number of labelled examples (otherwise known as the *ground-truth*) to the training algorithm. The algorithm uses these data to construct the classification rules required to label new previously unobserved data points. Supervised methods are generally much more accurate than unsupervised methods because labelling the training data makes it easier to generate accurate estimations of the distribution of values in the observations. The generation of classifiers through supervised learning takes one of two approaches; either a generative approach or a discriminative approach.

2.2.3 Generative versus Discriminative Classification

In the generative approach to classification, the learning algorithm seeks to model the distributions of the observations in the training data given the labels and therefore learn an estimate for the population distributions of the object types over the features being measured. This means modelling the likelihood $P(X | Y)$ and the prior $P(Y)$. Through Bayes' rule, the probability that some particular class label y_k applies given a feature vector measurement \mathbf{x}

can be written as

$$P(Y = y_k | X = \mathbf{x}) = \frac{P(X = \mathbf{x} | Y = y_k)P(Y = y_k)}{\sum_j P(X = \mathbf{x} | Y = y_j)P(Y = y_j)} \quad (2.1)$$

where \mathbf{x} is a feature vector having N dimensions. If (for calculation sake) it is assumed that \mathbf{x} is comprised of boolean values, this means that in order to be able to model the likelihood distribution $P(X | Y)$ using training data, at least one of every single combination of values of \mathbf{x} would be needed – and in fact very many more because the distribution would be modelling the frequency of the different combinations. To calculate the actual values of these frequency parameters (which must sum to one), only $2^N - 1$ parameters are needed. If two classes are being modelled ($|Y| = 2$), then this requires estimating $2(2^N - 1)$ parameters for the distribution. Typically, this is not feasible for even relatively short feature vectors of $N \approx 10$ since this would require more than $2(2^{10} - 1) = 2046$ labelled observations and most feature extractors generate much longer vectors with N in the order of 10^2 or more.

2.2.4 Naïve Bayes

For generative classifiers to be able to estimate the distribution of the likelihood function $P(X = \mathbf{x} | Y = y_k)$, some *a priori* assumptions need to be made about how the feature values in \mathbf{x} occur. The simplest assumption that reduces the complexity of modelling the distribution is to assume that the values of \mathbf{x} are conditionally independent of each other given the class label $Y = y_k$. This turns a computational problem that is exponential in N into a problem that is only linear in N . This naïve assumption about the correlations between the variables in \mathbf{x} allows for equation 2.1 to be rewritten as the Naïve Bayes classification function:

$$P(Y = y_k | X = \langle x_1, x_2, x_3, \dots, x_N \rangle) = \frac{P(Y = y_k) \prod_i^n P(x_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i^n P(x_i | Y = y_j)}. \quad (2.2)$$

The most likely classification is found by maximising this function over all possible values of Y where the denominator in equation 2.2 is ignored since it does not depend upon y_k :

$$Y \leftarrow \arg \max_{y_k} P(Y = y_k) \prod_i^n P(x_i | Y = y_k). \quad (2.3)$$

If the values that are encoded by the feature vectors \mathbf{x} are discrete (and not boolean), it is necessary to estimate the frequency of occurrence of the different values that each element of

\mathbf{x} can take on. Similarly, more than two classes may be modelled by the classifier and this requires estimating the parameter that specifies the prior likelihood of an observation being from a class $Y = y_k \in K$. Given some dataset H having observations $\langle X, Y \rangle$ that comprise a N dimensional feature vector $X = \mathbf{x}$ and a label $Y = y_k$, with each element $x_i \in \mathbf{x}$ being able to take on values from J (with $x_i = j \in J$), then a single parameter for that distribution can be found using *maximum likelihood* estimation as

$$\hat{P}(x_i = j | Y = y_k) = \frac{|\{\langle X, Y \rangle \in H | x_i = j \cap Y = y_k\}|}{|\{\langle X, Y \rangle \in H | Y = y_k\}|}. \quad (2.4)$$

That is, $P(x_i = j | Y = y_k)$ is estimated as the frequency of observations where the i^{th} element of a feature vector \mathbf{x} takes on the value j , from within the set of observations labelled as belonging to class y_k . It is often the case that insufficient observations are present in the training data to be able to estimate some of the distribution parameters. In this case, to avoid the parameter being estimated as having zero likelihood of occurrence, a base probability for every observation is introduced. This corresponds to a *maximum a posteriori* (MAP) estimate for the parameter where each observation is presumed to have an initial equal non-zero probability of occurring:

$$\hat{P}(x_i = j | Y = y_k) = \frac{|\{\langle X, Y \rangle \in H | x_i = j \cap Y = y_k\}| + l}{|\{\langle X, Y \rangle \in H | Y = y_k\}| + l|J|}. \quad (2.5)$$

This is known as *Laplace smoothing* where l is usually set to one. Similarly, the prior probability for the occurrence of an observation having a particular label $Y = y_k$ can be estimated as

$$\hat{P}(Y = y_k) = \frac{|\{\langle X, Y \rangle \in H | Y = y_k\}| + l}{|H| + l|K|}. \quad (2.6)$$

To estimate the parameters of the likelihood function $P(X = \mathbf{x} | Y = y_k)$ when $\mathbf{x} \in \mathbb{R}^N$, the most common method is to make a second assumption that the values that each individual element $x_i \in \mathbf{x}$ can take on are normally distributed such that $P(x_i, y_k) \sim \mathcal{N}(\mu_{ik}, \sigma_{ik})$. This results in the Gaussian Naïve Bayes (GNB) classifier.

2.2.5 Logistic Regression

In the discriminative approach, the learning algorithm seeks to model the posterior distribution $P(Y | X)$ from the training data directly. Discriminative approaches are appropriate when simply needing to determine class membership of an observation or when a simple classification function is needed. Linear classification boundaries are typical – especially in the binary classification problem – and are simple to represent; linear classification functions can

be encoded straightforwardly as vectors of weights.

The canonical example of a supervised learning method that can be used to perform discriminative classification is *logistic regression*. As in naïve Bayes, distribution parameters are assumed but instead of these being used to model the likelihood function $P(X | Y)$, the assumptions place constraints around the parameters of the posterior function $P(Y | X)$ and the ground truth observations are used to directly train these parameters. In the binary classification case such that $Y \in \{0, 1\}$, the models for the two classes are given as

$$P(Y = 0 | X = \mathbf{x}) = \frac{1}{1 + \exp(w_0 + \sum_i^N w_i x_i)} \quad (2.7)$$

$$P(Y = 1 | X = \mathbf{x}) = \frac{\exp(w_0 + \sum_i^N w_i x_i)}{1 + \exp(w_0 + \sum_i^N w_i x_i)} \quad (2.8)$$

where $P(Y = 0 | X = \mathbf{x}) + P(Y = 1 | X = \mathbf{x}) = 1$.

The weight vector $\mathbf{w} = \langle w_0, w_1, w_2, \dots, w_N \rangle \in \mathbb{R}^{N+1}$ represents the distribution parameters for these functions and it must be learned from the labelled training data. Note here that there are N parameters (one for each of the elements of \mathbf{x}) with an extra parameter which represents the prior probability of the class value taken on by Y . One reasonable approach to estimating \mathbf{w} is to select parameters that allow for the likelihood of the labelled training data to be maximised conditioned on these parameters:

$$\mathbf{w} \leftarrow \arg \max_{\mathbf{w}} \prod_l^{ |H| } P(Y^l | \mathbf{x}^l, \mathbf{w}). \quad (2.9)$$

This can be reformulated into a sum over logarithms as

$$\mathbf{w} \leftarrow \arg \max_{\mathbf{w}} \sum_l^{ |H| } \ln P(Y^l | \mathbf{x}^l, \mathbf{w}). \quad (2.10)$$

This conditional log likelihood can be written in the binary classification case as

$$\ell(\mathbf{w}) = \sum_l^{ |H| } Y^l \ln P(Y^l = 1 | \mathbf{x}^l, \mathbf{w}) + (1 - Y^l) \ln P(Y^l = 0 | \mathbf{x}^l, \mathbf{w}) \quad (2.11)$$

which takes advantage of the fact that in the binary classification case $Y \in \{0, 1\}$ and so only one of the terms in the summation can ever be non-zero for each value of Y^l . Substituting in

equations 2.7 and 2.8, equation 2.11 can be rewritten as

$$\ell(\mathbf{w}) = \sum_l^{|H|} Y^l \ln \frac{P(Y^l = 1 | \mathbf{x}^l, \mathbf{w})}{P(Y^l = 0 | \mathbf{x}^l, \mathbf{w})} + \ln P(Y^l = 0 | \mathbf{x}^l, \mathbf{w}) \quad (2.12)$$

$$\ell(\mathbf{w}) = \sum_l^{|H|} Y^l (w_0 + \sum_i^N w_i x_i^l) - \ln(1 + \exp(w_0 + \sum_i^N w_i x_i^l)) \quad (2.13)$$

Since a closed form solution to maximising $\ell(\mathbf{w})$ with respect to \mathbf{w} does not exist, the function must be optimised using numerical methods such as gradient ascent. The gradient for this function is a vector of partial derivatives with the i^{th} element being of the form

$$\frac{\partial \ell(\mathbf{w})}{\partial w_i} = \sum_l^{|H|} x_i^l (Y^l - \hat{P}(Y^l = 1 | \mathbf{x}^l, \mathbf{w})) \quad (2.14)$$

with w_0 requiring that each observation in the training data be extended with $x_0 = 1$. $\hat{P}(Y^l | \mathbf{x}^l, \mathbf{w})$ is evaluated during each iteration of gradient ascent according to the current estimate of \mathbf{w} using equations 2.7 and 2.8. The error in prediction is given by the difference between the labelled classification of Y^l and the currently estimated probability that the observation is labelled $Y^l = 1$. Note that in the binary classification case, this error term in the parentheses of equation 2.14 will be zero once $Y^l = \hat{P}(Y^l = 1 | \mathbf{x}^l, \mathbf{w})$ whether $Y^l = 0$ or $Y^l = 1$.

To perform gradient ascent on $\ell(\mathbf{w})$, at each iteration the elements of the gradient vector are updated according to

$$w_i \leftarrow w_i + \eta \sum_l^{|H|} x_i^l (Y^l - \hat{P}(Y^l = 1 | \mathbf{x}^l, \mathbf{w})) \quad (2.15)$$

with η being some small constant specifying the extent to which w_i is updated at the end of each iteration. $\ell(\mathbf{w})$ is a concave function in \mathbf{w} and so gradient ascent will converge to the function's global maximum.

This procedure can overfit to the training data especially in sparsely populated high dimensional datasets. One method that can be used to avoid overfitting is to replace function 2.10 with

$$\mathbf{w} \leftarrow \arg \max_{\mathbf{w}} \sum_l^{|H|} \ln P(Y^l | \mathbf{x}^l, \mathbf{w}) - \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (2.16)$$

where $\|\mathbf{w}\|^2$ specifies the ℓ_2 -norm of \mathbf{w} or the Euclidean distance described in parameter space by the weights vector. Known as *regularisation*, this modification has the effect of penalising

large variations between the individual weights and acts to give a more “rounded” distribution over all of the parameters – de-emphasising the impact of outliers in the training data. The constant λ is set to determine the impact of this penalty term. This results in the modified gradient ascent rule:

$$w_i \leftarrow w_i + \eta \sum_l^{|\mathcal{H}|} x_i^l (Y^l - \hat{P}(Y^l = 1 | \mathbf{x}^l, \mathbf{w})) - \eta \lambda w_i. \quad (2.17)$$

Generative classifiers construct distributions that allows a new instance’s class membership to be determined by evaluating its placement within the different class models (distributions) available to the classifier. Generative models maintain parameters that describe the distributions and so new observations can be incorporated into the existing models. As such, generative classifiers are more suitable for reinforcement (or semi-supervised) learning which is achieved by updating the likelihood function $P(X | Y)$ along with the prior $P(Y)$. It is difficult to incorporate new observations into existing models encoded by discriminative classifiers because the target function $P(Y | X)$ must be regenerated by reevaluating all of the existing observations against the new observation. This necessarily entails maintaining a copy of the features used to generate the classifier which greatly increases the amount of data needed to maintain the classifier.

In both approaches, the more data that are available, the more accurate the classifiers. Both approaches must avoid overfitting to the training data *i.e.* deriving a classification function that expresses too much bias towards the classification of observations that are very similar in nature to the training data used to generate the classifier. As shown above in the case of logistic regression (and other discriminative classifiers), a regularisation parameter can be incorporated into the parameter estimation function to reduce the impact of such bias.

2.2.6 Classification Accuracy: Recall and Precision

The metrics that determine the accuracy of the classification function are given by the true positive rate (or recall) and the positive prediction rate (or precision). This section discusses how these concepts are impacted by the nature of the classification scheme. The calculation of these metrics and how they are used to show differences in classification accuracy are detailed in section 3.5.

A feature extractor that gives good classification recall will not encode features about the objects of interest (the positive class) that only ever appear in “non-object” images (the

negative class). Optimising recall can be achieved by increasing the variance of the values encoded by the feature vectors extracted from the training data, such that more of the object’s “population” may be represented by the classifier, thus increasing its ability to generalise to new data.

A feature extractor that gives good classification precision encodes features about the objects of interest that are sufficiently dissimilar from non-object images so that very few negative class images will be mistakenly labelled by the classifier as positive class instances. Optimising precision can be achieved by decreasing the variance of the values encoded by the feature vectors extracted from the training data, so that *only* the objects given in the training data will be represented in the classifier. Unfortunately, increasing precision to this degree means that the classifier will overfit to the training data. While the classifier will be very precise in its ability to specifically exclude anything that is not labelled as a positive instance in the training data, it will have difficulty in being able to generalise to any new examples of the object type (*i.e.* classify as positive examples of the type examples that are not already included in the labelled training data).

The aim is to maximise both recall and precision in a classifier but because the mechanism by which this is achieved requires changing the allowed degree of variation in the feature values modelled by the classifier in opposite directions, the two concerns are in tension. The decision boundary given by the classifier (if using a discriminative classifier) (*i.e.* the thresholding hyperplane in feature space that delineates the identification of positive versus negative instances in the binary classification problem), will almost certainly intersect the population distributions of both the positive and the negative examples even if no intersection is present according to the samples used to train the classifier. With generative classifiers, this problem becomes one of determining the true membership of a test instance when the estimated population distributions of the positive and negative classes overlap (which is always the case when assuming distributions based upon Gaussian functions which never give exactly zero at any point in their domain – which could be over all possible feature values). For practical classification, thresholds must be used to determine when a given test instance is sufficiently on either side of the classification boundary in the discriminative classification case, or under a sufficient mass of the modelled probability distribution function in the generative classification case.

A very simplified depiction of the binary classification problem in the context of discriminative classification is given in figure 2-1 which shows a feature space having two dimensions with each point in this space in \mathbb{R}^2 . The complete population distribution of the positive examples within this space is shown as the blue region, and the complete population distribution of the negative examples is shown by the red region (in the binary classification formulation, the

negative class is simply any part of the feature space that isn't identified as being part of the positive class). The circles indicate the feature vectors used as training data (ground truth) for the positive class, and the crosses indicate the feature vectors used as training examples of the negative class.

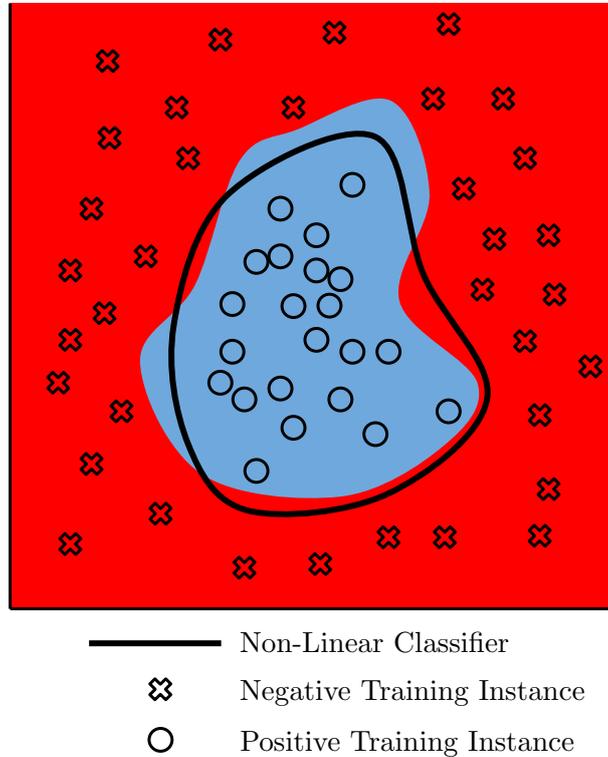


Figure 2-1: A simplified representation of the population distributions of two object classes Blue and Red within a two dimensional feature space, and the training data (circles and crosses) used to train a classifier represented by the decision boundary (thick black line) estimating the learned demarcation between the classes. The estimated boundary is an imperfect fit to the true boundary and this will invariably lead to errors in classification.

The two dimensional feature space slices through the actual population distributions which are much more complex in terms of the number of dimensions required to completely represent them. Given these training data, the discriminative classification function is designated by the black boundary which represents the learned estimation of the boundary between the two classes. The classification function specifies that anything inside the boundary should be labelled as a positive (blue) instance, and anything outside the boundary should be labelled as a negative (red) instance. This boundary is not exactly incident with the true boundary between the populations and so errors in classification will occur if new observations from the red area inside this boundary are evaluated. This will result in false positives, or Type I errors – causing a degradation in the precision rate while the recall rate is unaffected. Similarly,

false negatives (or Type II errors) will result if observations from the blue area outside the boundary are evaluated by the classifier (since these will be erroneously classified as red instances) – causing a drop in the recall rate while precision is unaffected.

In this idealised situation, there is a single continuous boundary segregating the two populations. In practice, there may be many viable decision boundaries between the populations given the specific dimensionality of the feature space. Alternatively, a boundary for some specific feature dimensions may be too complex to be described by the given classifier. The greatest change to the decision boundary can be brought about by changing the nature of the feature space itself, either through the selection of different features, or by changing the dimensionality of the feature space. Therefore, “better” features have the desired effect of simplifying the representation of the “true” population boundary, increasing the potential of the learning algorithm to estimate this boundary given a fixed number of training data and *a priori* assumptions about the model being used to describe the boundary. The chosen learning algorithm itself is constrained in how it can represent a decision boundary (or a population distribution in the case of generative classifiers) and so the nature of the features extracted in the observations being used to train the classifier tend to have greater significance in controlling the ability of the learning algorithm to find a suitable classification function. More succinctly, while the learning algorithm determines how a space is classified, the feature extraction algorithms determine the nature of the space being classified.

The choice of features constrains the classification function in being able to operate only in a certain feature space. Representational variance exhibited by other features (in other dimensions) cannot be learned. This problem can be mitigated by ensuring that the features being extracted for each observation are capable of capturing the expected variation in observations from the query data. In classical object recognition, objects are represented in appearance in terms of RGB or intensity images and so only appearance based features can be used to model the expected distributions of the object classes. The use of multi-modal data – range information in this case – can expand the domain of features available for use in modelling the representational variance of the object populations. Features that encode depth information may allow for other object characteristics to be modelled in the classifier (*e.g.* surface morphology and silhouette/boundary information). These features are explored more with a view to improving classification accuracy in chapter 4.

When modelling different features in a classifier, the range of variance in the different feature values that are extracted must be standardised (usually between zero and one) so that the training algorithm does not weight certain features as having greater significance than others (some features will inherently express variation over a broader range of values than others). The specific training algorithm may require that the feature vectors are processed further

– normalisation using the ℓ_2 -norm (Euclidean distance) is a common stipulation for many classifiers.

Finally, many learning algorithms undertake dimensionality reduction on the extracted feature vectors (such as PCA) before using these reduced feature vectors in the training of the classifier itself. This is because classifiers may use the same feature extraction algorithms to collect observations over a number of different object categories. These feature vectors can express correlated variances in their elements in different ways depending upon the object type. Understanding which of these covariances are important to model can increase the accuracy of recall since basis vectors in the feature space along which less variance is represented may be indicative of spurious features for that particular object class. In addition, the complexity of the classifier itself is reduced making for more efficient processing. In this thesis, while multiple object types are classified, the use of dimensionality reduction techniques is not investigated since the focus is on evaluating the performance of other aspects of the object recognition pipeline.

2.2.7 Combining Classifiers

Classifiers can be combined together in several ways to enhance accuracy. These methods include *bagging* (short for *bootstrap aggregating*) which is a form of *ensemble* learning involving the use of multiple independently trained classifiers. These are constructed by sampling (with replacement) from the training dataset to produce classifiers that all behave slightly differently, but provide aggregated decisions (by majority vote) that are more representative of the populations from which the ground truth observations are sampled. The effect of classifier bagging is to further reduce the influence of unrepresentative data by creating a selection of different classifiers each having their own individual biases. These biases ought to be randomly distributed and so they act to cancel one another out when taking an average result.

Distinct from classifier bagging is *boosting* which combines together many poor quality classifiers trained using small subsets of the ground truth data, but with weights iteratively trained and associated with each subset of the training data to influence the behaviour of the classifier regarding the value distribution of certain features. Boosting requires a much closer integration with the classification scheme being employed and is generally implemented within the recognition scheme rather than as a *post hoc* addition.

2.2.8 Classifiers Used In This Thesis

How object recognition is undertaken is largely determined by the specific nature of the task. The schemes in use today are those geared toward the classification of objects from predetermined categories using state of the art machine learning classifiers such as *Support Vector Machines (SVMs)* (Felzenszwalb *et al.*, 2010; Zhang *et al.*, 2011; Gao *et al.*, 2011; Ozuysal *et al.*, 2009; Zhang and Chen, 2010; Bo *et al.*, 2011; Dalal and Triggs, 2005), *Random Forests* (John *et al.*, 2010; Schroff *et al.*, 2008), and *Neural Networks* (Rowley *et al.*, 1998; Krizhevsky *et al.*, 2012; Zhao and Thorpe, 2000; LeCun *et al.*, 1989; Cireřan *et al.*, 2011) amongst others. Marsland (2009) gives a good overview of several supervised and unsupervised techniques. Many recognition schemes based on these methods (particularly neural networks) are experiencing a resurgence in popularity as Moore’s law (Moore, 1965) continues to hold and the processing power of modern computer technology increases exponentially allowing for ever more complex algorithms to be executed against ever growing datasets.

SVMs, random forests, and neural networks are all examples of discriminative classifiers. One possible reason for the increasing popularity of these kinds of classifiers (over generative classification methods) has to do with the availability of larger datasets. Ng and Jordan (2001) showed theoretically and empirically that because of the stricter assumptions made by generative classification methods concerning the distribution of observable features (for reasons of tractability), these methods do well with smaller datasets where the underlying assumptions in the model more accurately reflect the true distribution parameters of the features. However, in datasets where these naïve assumptions reflect the true distribution parameters less accurately, or where more ground truth data are available to train the classifiers, discriminative classifiers demonstrate greater accuracy.

In this thesis, two different supervised machine learning algorithms are used. Due to its recent popularity and performance in a large number of other studies (Osuna *et al.*, 1997; Pontil and Verri, 1998; Maldonado-Bascon *et al.*, 2007; Maji *et al.*, 2008; Felzenszwalb *et al.*, 2010), the classification scheme used in chapter 4 to help compare the accuracy of two different feature extraction techniques is the SVM (Schlkopf *et al.*, 1995; Burges, 1998).

In chapter 5, an existing machine learning technique called a *Hough Forest* (based upon random forests) is modified to make use of depth information available in its training data, and the accuracy afforded by the adjusted technique is compared against the original implementation in the context of a parts-based object detection scheme. This method is selected for its ability to aggregate independently detected object parts into a whole object detection scheme. The learning method is interesting because it alternately invokes classification (to compare

the similarity of object parts) and regression (to determine their likely location relative to the comprising object).

2.2.9 Support Vector Machines

The linear SVM is a discriminative classifier that finds a set of “support vectors” and associated weights to parameterise a hyperplane in feature space that separates the positive and negative class instances. The support vectors are so called because they are the minimum set of training data that are needed to define the separating hyperplane. For a linear classifier, the size of this set is equal to the number of feature dimensions. The boundary surface is a flat plane so it can be defined in terms of a single gradient vector orthogonal to the plane with the same number of dimensions as the feature space.

In the standard SVM algorithm, a set of weights are trained – one per support vector – using an iterative gradient ascent method akin to logistic regression. In the linear classifier, these weights decide the orientation of the gradient vector in each dimension. This greatly simplifies classification which then only entails the evaluation of the dot product of the plane’s gradient vector with the feature vector from a new observation in order to find the distance of the new example to the plane, and on which side of the plane it occurs. The downside to the linear classifier is that the feature encoding used may not lend itself to a clean separation of the positive and negative examples. In this case, a more complex separating surface is required which is why non-linear kernel functions are used.

Figure 2-1 depicts feature space with two dimensions where the separation between the positive and negative instances requires a boundary that cannot be described using a simple linear function. Figure 2-2 shows another representation of the two class situation where a different feature extractor has been used to define the feature space. The use of different features results in a different separation between the two classes. In this case, a linear classifier can be used (shown as the solid black line) to separate the training data (shown as the crosses and the circles) provided to the SVM algorithm. Given the ground truth observations, a non-linear separating boundary given by the thick dashed line might also be possible. In this case, the non-linear classifier estimates a function that separates the positive and negative class populations more accurately (given by the boundary between the blue and red regions).

A non-linear classifier can give improved accuracy but the information needed to define the separating boundary cannot be stored as concisely. Non-linear functions such as polynomials

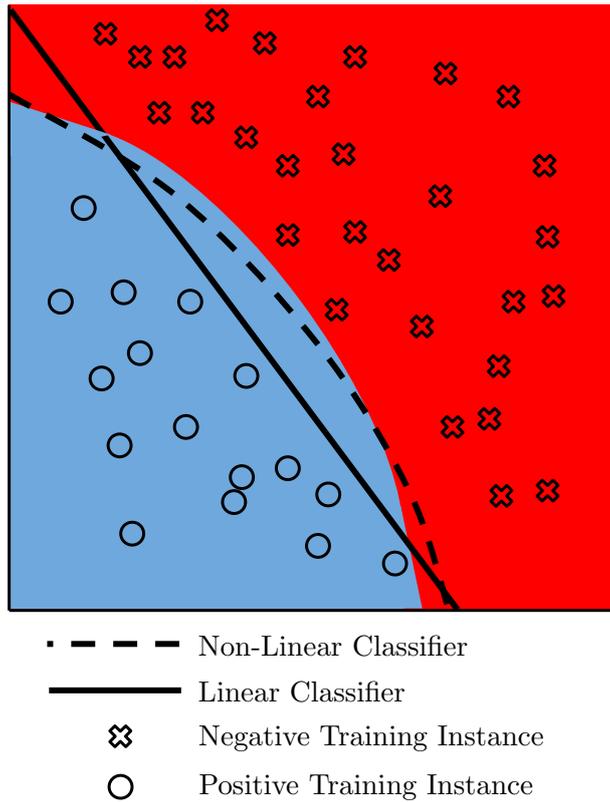


Figure 2-2: The positive (blue) class and the negative (red) class population distribution within a two dimensional feature space is shown. Both the linear and the non-linear classifier fit the data, but due to the more accurate fit to the true population boundary given by the non-linear classifier, it offers greater precision and recall accuracy.

or exponential functions require that the classifier store the support vectors and evaluate the test feature vector against each of the support vectors as a weighted sum of the kernel function. This can significantly slow down classification unless “good” parameters for the function are found. Good parameters tend to result in a smaller number of support vectors being needed to define the boundary between the positive and negative examples. Fewer support vectors should be preferred to avoid overfitting to the training data, but not so few that the boundary cannot model enough of the true variance in the population distributions of the data.

In this thesis, both linear and non-linear SVMs are used. The non-linear classifier uses a *radial basis function* $RBF_{\gamma}(\mathbf{x}, \mathbf{s}, \alpha)$ for some support vector \mathbf{s} , its trained weight α , and observation \mathbf{x} . It is defined as

$$\boldsymbol{\delta} = \mathbf{x} - \mathbf{s} \tag{2.18}$$

$$RBF_{\gamma}(\mathbf{x}, \mathbf{s}, \alpha) = \alpha \exp(-\gamma \boldsymbol{\delta} \cdot \boldsymbol{\delta}) \tag{2.19}$$

Parameters γ and α should be chosen empirically through cross validation so as to give a good fit to the training data. The non-kernel specific cost term α influences the nature of tolerance that the classifier has toward misclassifications and can be used to decrease the proportion of false negatives at the cost of increasing the false positive rate. Alternatively, a lower false positive rate can be traded for an increased false negative rate.

The cost parameter is applied to all of the support vectors equally and changing it acts to shift the position of the separating boundary in feature space as it is specified by the support vectors. If a good separation isn't possible, modifying α changes which of the data are classified as positive or negative without changing the respective weighting of the false positive and false negative rates. For the linear classifier, α acts to translate the boundary in feature space. For radial basis functions, α acts to increase or decrease the volume within the separating boundary surface.

For this thesis, a custom highly concurrent version of the SVM training and classification algorithm was implemented in order to implement different kernel functions and to efficiently process the large amount of data required for the experiments undertaken. All code is written in C++, making use where possible of routines from the OpenCV¹ image processing and Boost² C++ extension libraries.

2.3 Object Modelling

Object modelling for classification concerns the manner in which features and other data about the objects of interest are stored within the recognition system, and how these models interact with the rest of the detection and classification pipeline to effect recognition.

In *bag-of-features* (or *bag-of-words*) models, features are encoded as *global* characteristics of an object. No spatial information concerning the relative placement of the features in the object is encoded. A feature vector represents a collection of measurements over an object (or a subregion of an image). Each element of the feature vector specifies a particular kind of measurement, but the measurement itself is calculated over the whole of the object. The feature measurements as they are stored at different locations in the vector may or may not be independent of feature values at other locations in the vector given the object type.

A rigid descriptor encodes measurements concerning *localised* features of an object as it

¹<http://opencv.org>

²<http://www.boost.org>

appears from a particular viewpoint and the ordering of the features in the descriptor can depend upon how the objects are represented in imagery. Rigid descriptions can be very susceptible to small changes in appearance caused by variance in orientation, lighting, and object morphology (especially in the case where objects are articulated). A rigid descriptor makes assumptions about the values of the elements in certain positions of the vector given a particular object type; a particular element in the vector gives a measurement for some feature at some specific location in the object.

Bag-of-features classification schemes have been effective in some unsupervised object classification problems (Sivic *et al.*, 2005), but such models are limited in their ability to discriminate between objects. While the values of certain global characteristics can be important, the correlations between local characteristics of an object can provide the classifier with useful discriminative information. With descriptors that encode features from specific locations, the values of the individual elements in the vector are often dependent upon one another; the presence of a feature at some location often correlates strongly with the values of features at other locations in the object (other elements of the descriptor). Rigid descriptors are often too inflexible to account for the wide range of variance innate to an object type unless the view of the object type can always be well constrained. If the view cannot be well constrained, models that can tolerate the required degrees of variance in the object's representation must be used.

One way in which the effects of representational variance can be mitigated is to use interest point detectors which can help to identify features that are more salient about an object and that are more resilient to the problems of object representation. For example, the Harris (Harris and Stephens, 1988) and FAST (Rosten and Drummond, 2006) detectors can be used to detect corners in an image which are typically more robust to slight changes in appearance (and are typically rotationally invariant) or the SIFT (Lowe, 1999) keypoint detector which allows for the scale invariant detection of keypoints. All these interest points are described by relatively simple features that require a small number of values to characterise. The reason for this is that they are typically combined together in recognition schemes so that many features of the type when taken together can indicate the presence of an object of interest. If a geometric model of a specific object instance is available then recognition can often be carried out by looking for a correspondence of the detected keypoints with the locations of the keypoints as they are known on the object model template. This geometric template matching approach is not very extensible to different object categories, or objects that have a high degree of intraclass variance. Without any kind of prescribed relationships between the detected interest points, the mere detection of such keypoints by themselves is typically not enough information to determine if an instance of a particular object type is present; many different object types can exhibit the same kinds of simple keypoints.

By combining the presence of simple features together with their relative positions, objects can be modelled with greater discrimination. The challenge is in how these models can be used for recognition. In a purely probabilistic classification scheme (such as Naïve Bayes), this would require constructing and evaluating probability distribution functions for every combination of basic feature – even those combinations that are not possible must have some small probability attached to them. More intelligent modelling recognises that the presence or values of certain features is conditioned only on a small subset of other features. A generalised learning scheme must know which of these features should be conditioned on which others.

2.3.1 Modelling Objects by Parts

Exploiting the spatial relationships between object parts has long been a popular approach to object recognition (Nevatia and Binford, 1977; Fan *et al.*, 1989; Sengupta and Boyer, 1995; Dickinson *et al.*, 1997; Huber *et al.*, 2004).

Models than encode the relative locations of the parts of an image or an object are known as *pictorial structures* (Fischler and Elschlager, 1973; Felzenszwalb and Huttenlocher, 2005). These models represent objects as a deformable configuration of their individual parts. The individual object parts encode the localised appearance of the object. The deformability of the model of the object arises from a functional specification of the geometric relationships between the parts so that configurations that occur more frequently are given greater weight. In object recognition, these models are known more generally as *deformable parts models* (Felzenszwalb *et al.*, 2008). The models can also be combined into hierarchies where they are generalised as *grammar-based models* (Zhu and Mumford, 2006).

Recognition schemes that use a parts based model framework proceed in two phases. Firstly, the initial detection of the individual parts (the simpler features) that could comprise object instances is carried out. Secondly, an analysis of the configuration of these parts to one another is undertaken. One of the more popular methods of analysing the relative positions of the object parts is by applying a *distance transform* to the locations of the detected parts. This image processing function measures the distances between specific locations in an image. Once the distances are known, the relative angles between the features/parts can be estimated. If the parts are detected in an acceptable configuration (within the parameters of the model), a strong detection response can be obtained. Calculation of the Euclidean distance transform can be computationally expensive and several schemes use generalised distance transforms instead. A detailed discussion of the Euclidean distance transform together with

some optimisations of the state-of-the-art algorithm is presented in chapter 6.

Parts based object recognition schemes have become popular in recent years because of their good recognition performance (Gall *et al.*, 2009; Kumar and Patras, 2010; Rematas and Leibe, 2011; Barinova *et al.*, 2012; Felzenszwalb *et al.*, 2010). One problem with parts based models is in how to learn the relationships between the object parts. If training data of an object type are provided with the individual parts already labelled, then it is simple to learn distributions of these parts in relation to one another. However, it is normally the case that training data are labelled with object identifiers without any more detailed sub-labelling of the individual parts. In these cases, it is necessary to automatically determine the salient parts of the object instances and to categorise these parts according to their similarity before a parts based model can be learned. While this task can be carried out using unsupervised learning approaches (such as k means, or k nearest neighbours), because the categorisation is carried out only on the appearance of the parts, their relative spatial locations are not taken into account. This can result on parts of the object being modelled with too much flexibility in their allowed placement which diminishes the utility of modelling the spatial relationships between the parts.

One way of addressing the issue of which parts to model while at the same time constraining their relative spatial positions, is to stipulate the topological dependencies between specific parts whether or not such dependencies really exist. One popular topology of parts is a star topology which makes the presence of the individual parts contingent only on the presence of a “root” part. This reduces the combinatorial complexity of the modelling problem (each part is only conditioned on the presence of one other part), while still enforcing strict relationships between the parts through the presence of the root part. The relative positions of each part with the root part can also be modelled using this arrangement. These topological arrangements of parts are both simple to model and allow for straightforward computability. As with grammar-based models in general, such models can be comprised of multiple levels with a different root part specified at each level with parts encoded at different resolutions (Felzenszwalb and Huttenlocher, 2005; Fidler and Leonardis, 2007).

Parts-based approaches to object modelling and recognition have shown superior performance to global appearance or bag-of-features based methods in several ways. Parts-based recognition approaches show improved robustness to variances in object representation as well as where objects are partially occluded where the detection of individual parts can be used to infer the presence of the whole. This can also allow for the detection of an object’s pose (position and orientation) due to the part correspondences (Sun *et al.*, 2010). Part-based approaches can result in decreased training and memory requirements. If parts are simple enough, they can be shared between multiple object types and the detection of specific parts reduces the

set of possible object types in a multi-class classification scheme (Ott and Everingham, 2011). In a multi-class recognition scheme, this approach can also be faster since unrelated object types are not tested for sequentially (linearly). A hierarchical approach to detecting the parts that comprise the objects of interest (*i.e.* in a grammar-based approach) can logarithmically pare down the size of the set of object type candidates (Mikolajczyk *et al.*, 2006; Ott and Everingham, 2011). Finally, for object types that express a high degree of intraclass variability, either where the object types are articulated, or simply where parts do not always have strict spatial correspondences, parts-based approaches are better able to represent the innate degree of variability in the type (Felzenszwalb and Huttenlocher, 2005).

2.3.2 Implicit Shape Models

The main drawback with parts-based models is the need to define *a priori* the nature of the relationships between the object parts *i.e.* the topology. For some object types, it is possible that even a very general topology (such as the star topology) does not provide a good model for the parts of the object. *Implicit Shape Models* (ISM) (Leibe *et al.*, 2004, 2007) arose from extending the *Hough Transform* (Hough, 1962) to understand how the detection of simple features could be used to infer the presence of more complex objects (Ballard, 1981). ISMs build dictionaries (or *codebooks*) of descriptors for individual object parts. These descriptors are generated from the locations of interest points detected in training examples. The spatial distributions of the part descriptors are defined in terms of their relative positions from the centroids of the whole object instances they are generated from. In this way, ISMs model the individual object parts along with their spatial relationships using an implicit star topology. While the centres of the object are usually used as the point of reference for the part offset vectors, any predefined position relative to the object can be used. For simplicity in the remainder of this section, the centroid of the objects will be assumed as the object reference point.

Figure 2-3 shows how ISMs are constructed from training examples to produce part codebooks for two different abstract object types – jagged “explosion” shapes, and rectangular shapes with rounded corners. Since the only two salient characteristics of these objects is their boundaries, their shape is indicative of their “object type”. The “parts” extracted from around the boundary of the objects (indicated by the black squares) represent the local descriptors that will be used to train classifiers to detect those parts in new (query) images. Each part has associated with it an offset vector pointing to the position of the object’s centre (the red dots). The part descriptors (labelled here with letters for clarity) when associated with the object centroid offsets and the class label of the object from which they are extracted

make up the codebook entries shown in table 2.1. Note that the codebook shown is generated from two single examples. Multiple examples are normally used to generate the codebook, and the offsets for each of the codebook parts determine a distribution of votes relative to the part rather than a single vote location given by single vector.

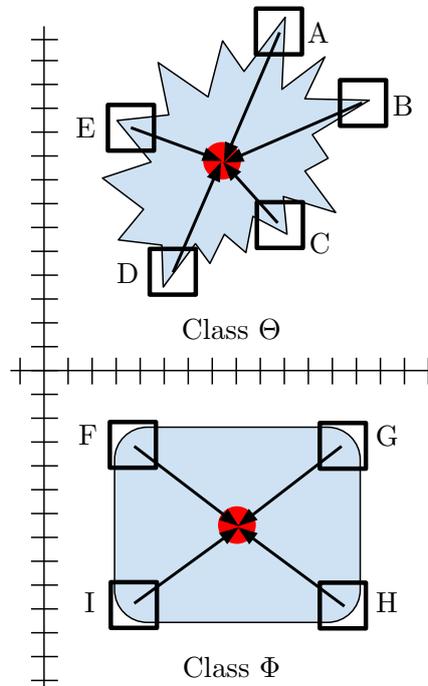


Figure 2-3: Extracting “part” descriptors as part of the codebook generation process for two ISMs encoding “jagged explosion” and “rounded rectangle” object types.

When being used for the detection of objects in a query image, the detection of the individual parts is carried out in a transformed *Hough* space where the relative 2-D offsets of the detected parts to their comprising object centroids is added to the 2-D position vector of the detected part in the query image. This means that many independently detected parts “vote” for the same object centroid in Hough space. The part classifier has access to the models of multiple part “types” from the same object class through the generated codebook. Upon classifying a sub-image extract as a particular type of part, the associated offset is used to determine the relative location of the object’s centre. In this way many different parts from the same object can be detected and the sum of the position vectors of these parts plus their object centroid offset vectors will result in the same position vector denoting the centroid of the object they comprise. This point in Hough space will receive a large number of votes – one for each detected part. A large number of votes at one position in Hough space therefore denotes the likely presence of an object comprised of these parts.

Part	Class	Offset
A	Θ	-2.5, -5.0
B	Θ	-6.0, -2.0
C	Θ	-2.5, 2.5
D	Θ	2.0, 5.0
E	Θ	4.0, -1.5
F	Φ	4.5, -3.5
G	Φ	-4.5, -3.5
H	Φ	-4.5, 3.5
I	Φ	4.5, 3.5

Table 2.1: An example codebook of parts from two different object types (Θ and Φ) based on the extracted parts of the “training” examples in figure 2-3.

This is represented in figure 2-4 which uses the parts extracted from figure 2-3 and stored in the generated codebook in table 2.1 to detect new instances of the object types. Even though the objects vary from the training examples in their global appearance, they are similar in their local part characteristics and so these parts, when detected independently, vote for common object centroids. Even though the votes by the individual parts do not exactly agree, they cluster closely enough to infer the presence of the comprising objects.

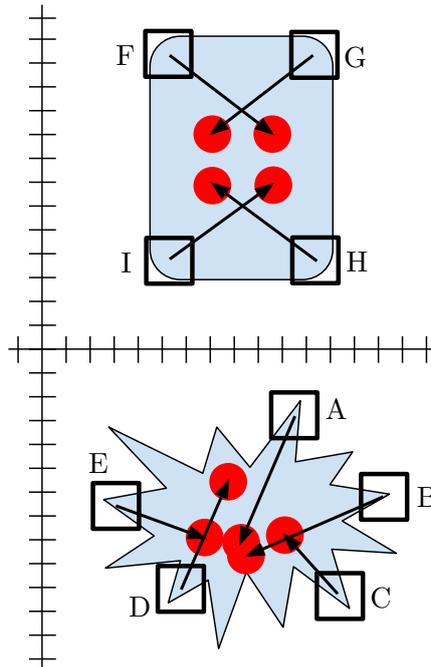


Figure 2-4: The codebook of “parts” in table 2.1 can be used to detect objects having similar parts to those in the codebook.

One issue with this method of indirectly modelling and detecting the centre of an object is

that due to intraclass variance as well as forms of representational variance, the offset vectors (which are learned from training examples) do not exactly specify the object centres but rather the positions of the centres of the objects used to train the ISM. In the generated codebook for the ISM, the part offsets are samples from a 2-D distribution that describes the likelihood of the occurrence of the object centre given that part. Some parts are located at multiple positions on an object *e.g.* the wheels on a profile of a car. In this case, the same part will have associated with it a distribution of offset vectors to the centre of the object type that is described by a *tri-modal* 2-D distribution. This is because only two wheels are visible when viewing the profile of a car and the front wheels are usually indistinguishable from the rear wheels. A wheel “part” can be offset from a car’s centre in two different ways depending on where it is located, therefore the separate detection of the wheels will cause votes to be cast for the object centre in two different places. Only in one of these places – the actual car’s centre – will the votes from both wheel offsets coincide. This location includes the largest number of votes for the object centroid even though there are two candidate locations to the left and right of the wheels, each having fewer votes.

This situation is depicted in figure 2-5 where the independent detection of the wheels of a car (shown as the white circles) are used to infer the presence of the car itself. Three training examples have been used to generate this ISM’s codebook, but since each training instance has two wheels “parts”, each car instance gives two different offsets for the centroid based on the fact that the two wheels in each example are similar enough in their appearance that they are learned as a single part by the clustering algorithm that generates the codebook. For the detection of the depicted car, this means that there are three main locations in the image where the wheels are “voting” for the location of the car’s centre. The wheels of a car are relative to the centres of the training instances in two different relative locations and so twice as many votes are cast for the actual centre of the object as are cast at either of the other two locations – ahead of the car, or behind it.

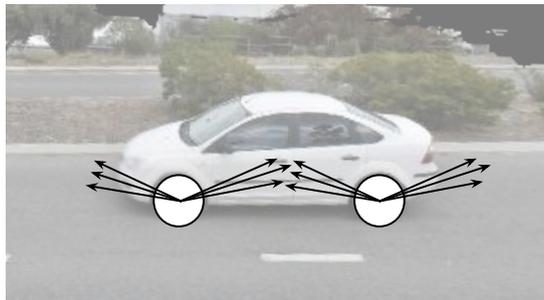


Figure 2-5: Detected wheel “parts” encoded by an Implicit Shape Model cast votes for the location of a car’s centre.

This situation is further complicated if objects of the type encoded by the ISM are not rotationally invariant. If the car in the query image is rotated, then given the same ISM, the wheel parts no longer agree on the centroid of the object since the part offsets do not take into account possible part rotations. This situation is depicted in figure 2-6. In general, the

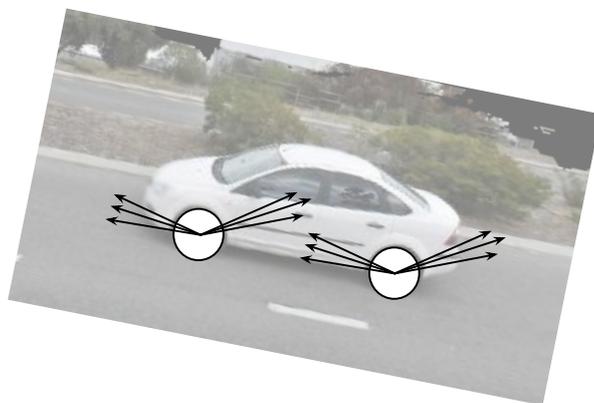


Figure 2-6: Due to the car’s rotation, the part offset vectors to the object centroid (which are derived from examples without the same degree of rotation) no longer agree and so the centre of the car is less reliably detected by the wheel “parts”.

remedy for the issue shown in figure 2-6 is to make the part offset vectors dependent upon the rotational orientation of the parts (the parts can be extracted using rotationally invariant descriptors). However, in this particular case, the rotational symmetry of the part, a wheel, makes it impossible to relate the orientation of the offset vector to the detected orientation of the part and it is necessary to control rotational variation by some other means. Even though parts such as wheels can present this kind of problem, most objects are not comprised solely of such parts and so they can be modelled in the ISM in a rotationally invariant manner that allows for the direction of the offset vector to be determined according to the detected orientation of the parts in the query image.

The voting mechanic used by ISM requires a discretisation of the continuous distributions described by the part offsets. Increasing detection precision requires a less coarse discretisation, but this increases the relative sparsity of votes in Hough space making it harder to determine the true modes and thus the likely positions of the objects. That is, there will be many more candidate detections at more definite locations, but each detection location will have associated with it a lower certainty. Alternatively, the resolution of Hough space can be decreased, increasing the relative density of votes at each location, but at the cost of decreasing detection precision.

A subsequent convolution of the Hough voting space with a Gaussian kernel of sufficient dimensions can help to “smooth” out the votes especially if a Hough space with relatively

high dimensions is used. The modes of this convolved Hough space then indicate the locations of the detected objects. If a kernel of insufficient dimensions is used, the multiple modes that indicate a single object detection will remain. Suitable dimensions for the kernel can be empirically determined so that these multiple modes are merged into a single local maximum. The dimensions of the kernel are partially dependent upon the innate characteristics of the object type. A type with high intraclass variance will require a Gaussian kernel with larger dimensions to account for the increased uncertainty in the relative offsets to the type's average centroid. Convolution with larger dimensioned kernels can be costly to compute and so it is necessary to balance the dimensions of Hough space with the dimensions of the Gaussian kernel.

Object Recognition Using ISMs

Implicit shape models themselves do not dictate any particular method of learning or classification. However, the need to model multiple different part “types” can be considered equivalent to the need to model multiple different simple object types. Classification methods must then allow for the discrimination between several different categories of object or part. Sequentially testing for the presence of each object type can be very inefficient if there are a large number of objects. It may also be difficult to accurately discriminate between similar object types (types that express low inter-class variance). Decision trees are a more efficient method of performing object classification over a large number of types. Decision trees partition feature space along different dimensions. The final classification decision is arrived at after multiple sequential decision stages where successive subregions of feature space are rejected according to the features being evaluated at each stage. The final classification associates the evaluated feature vector with the remaining non-rejected subregion of feature space. The object type(s) that populate that space are then the most likely candidates for the classification of the object. Since multiple object types can populate the same subregion of feature space, this allows a probability to be attached to an object classification based on the relative proportions of the different object types in the remaining subregion.

Classification schemes based upon decision trees are non-linear because even though each decision *stage* is itself linear, there are multiple linear decision stages acting orthogonally to one another in feature space which cause different subregions of feature space to be excluded at each stage. Figure 2-7 shows a two dimensional feature space that is partitioned according to different decision stages. The order of these stages specifies the decision boundary locations (shown with red lines). The decision tree classifier is able to find a sequence of decision boundaries so that the three different object types (the blue regions that estimate their population distributions in feature space) are roughly partitioned. While the partitioning may

not be perfect (as shown), the three partitioned spaces C_0 , C_1 , and C_2 contain a majority of one particular object type and these proportions estimate the likelihood of each space containing an instance of any particular object type.

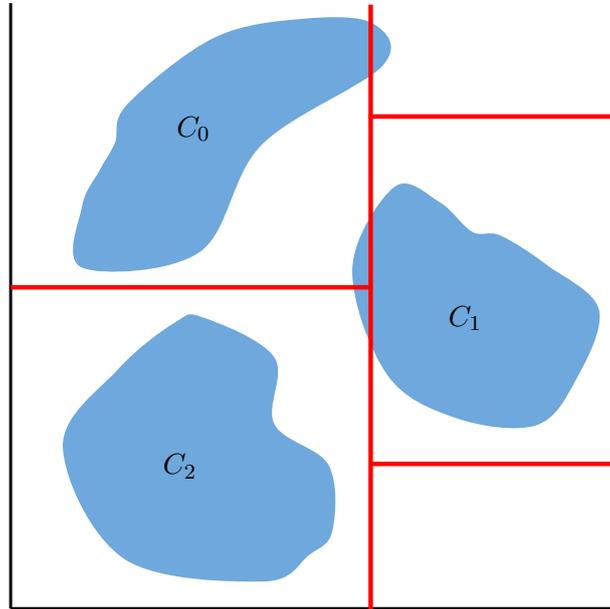


Figure 2-7: A partitioning of feature space by a decision tree that approximately matches the regions populated by three different object types (the blue areas).

The *Class Specific Hough Forest* (CSHF) (Gall *et al.*, 2009) is a refinement of the Hough forest based method that detects objects using an ISM. The CSHF differs from the basic Hough forest algorithm in that instead of learning codebooks of object parts, the CSHF directly associates part encodings extracted randomly from the training data to the object type. This avoids the need to undertake unsupervised learning (clustering) on the extracted parts and increases the computational speed with which object parts can be compared at detection time.

The appearance of the object parts along with their offsets can be trained in a combined classification/regression approach that employs an ensemble of decision trees called a *random forest*. At each stage in the decision tree, a random choice is made as to whether to optimise a classification metric based on the appearance of a part, or a regression metric based on a comparison of a part's offset from the centre of its comprising object. At detection time, feature descriptors generated from image patches are parsed by the decision tree to lookup the leaf nodes that represent the classification and regression estimations of the ISM encoded by the decision tree. A number of these *randomised decision trees* are used and the classification/regression decisions of each tree are aggregated into the final Hough voting

matrix. Together, this framework is termed a *Randomised Hough Forest* (Moosmann *et al.*, 2008).

The CSHF was developed originally as a 2-D object detection scheme. In chapter 5, several extensions to the CSHF recognition scheme are developed and tested that utilise the explicit availability of depth information in 2-D scenes to improve the accuracy of object detection.

Chapter 3

Datasets

This chapter gives an overview of the two main mobile mapping datasets used in this work, and the techniques employed in their generation. Both of these datasets provide depth alongside colour information, but the data are produced using different methods. The process of ground-truthing and how it was undertaken on these datasets is also described.

Section 3.1 briefly outlines the main techniques used to acquire the data used in this thesis. Section 3.2 describes the characteristics of the Earthmine dataset which was captured using stereophotogrammetry. Section 3.3 describes the AAM dataset and how these data were collected using a combination of laser scanning and colour imaging. Section 3.4 details the process undertaken to ground truth objects in both of these datasets within a common framework. These ground-truthed data are used to evaluate the accuracy of the object classification and detection techniques presented in chapters 4 and 5. Section 3.5 defines a method for evaluating classification and detection accuracy within a common mathematical framework.

3.1 Dataset Fusion

Traditionally, object recognition has been undertaken using data collected from single sensor modalities (*e.g.* 2-D colour images). In recent years, methods such as stereographic imaging and laser scanning have allowed for the explicit collection of spatial information about scenes (geolocations) alongside 2-D colour views. By coregistering the spatial information with the 2-D scene pixels, scene elements can be understood in terms of their depth relationships. This fusing of these data generated by different sensor modalities can potentially help in improving several aspects of object recognition.

The spatial data are collected in conjunction with information on the position and orientation of the sensing viewpoints by way of ancillary inertial tracking systems and Global Positioning Satellite System (GPSS) receivers. This provides spatial and contextual understanding of a scene that may be further exploited to assist in object recognition even while the nature of scene elements themselves are unknown. In an example of how combining sensor modalities can help with a simple object recognition task, Baatz *et al.* (2010) used mobile phone location data (collected via GPS and wireless tower triangulation) combined with separately obtained 3-D building models to re-orient images taken from a user’s mobile phone camera into street facing 2-D views. Standard 2-D computer vision techniques were then used to recognise buildings of interest. More broadly, the vast majority of computer vision and pattern recognition algorithms have been developed to specifically process 2-D image data. There are thus significant research opportunities to return to these algorithms with a view to exploring how they can be further developed to exploit multi-sensor fused datasets.

Three-D datasets of expansive urban environments have only recently become available due to the decreasing costs and complexity involved in the collection of such data using mobile mapping vehicles, and the improving accuracy with which a scene’s spatial information can be registered against its colour imagery. There remain challenges however, and the two main methods of collecting colourised spatial information (coloured point clouds) both have disadvantages – particularly when used in mobile mapping where the movement and vibration of the vehicle and its sensor arrays can disrupt the accuracy with which the data are acquired.

In stereographic imaging (or stereoscopy), scene data are acquired from multiple onboard 2-D cameras having known baseline offsets to one another. These images are stitched together and the slight differences in the 2-D image projections can be used to derive distance information and thus spatial locations. While colour acquisition is good, the spatial accuracy afforded by stereoscopy decreases with the distance to the scene elements. Post-processing (such as bundle adjustments) must be undertaken to try to mitigate point cloud registration errors with the 2-D colour image data. The Earthmine dataset, described in section 3.2, is generated in this fashion.

In laser scanning, spatial information about scene elements is acquired via the direct measurement of distance using lasers. Given the known constant speed of light, accurate measurements of distance are derived by timing the reflection of these lasers from scene elements back to the sensors located onboard the mobile mapping vehicle. Multiple measurements of a scene must be made to generate accurate data (due in part to the different refractive indices of the materials that the lasers are being reflected off of) and this generally requires several passes by the mapping vehicle (stereographic imaging requires only a single pass and so is more efficient).

The lasers are only able to acquire depth information and so separate camera sensors must be used to capture colour information about the scene. This must be coregistered against the depth information acquired by the lasers in a post processing step.

The different sensor modalities are calibrated to produce output that closely aligns, however slight differences in calibration and sensitivities to different environmental effects, as well as vibrations between the sensors can cause misalignments between the differently generated data. In addition, issues that are particular to one sensor (such as perspective effects in the case of the camera sensor) can further complicate the data fusion process. These issues result in inaccuracies in the registration of the colour information with the 3-D spatial data that can be difficult to entirely resolve. The AAM dataset, described in section 3.3, is generated in this manner.

3.2 The Earthmine Dataset

Earthmine¹ (now owned by Audi, BMW, and Daimler²) specialise in the collection, generation and delivery of street level 3-D imagery through the use of proprietary vehicle mounted 2-D image capturing systems, post-processing software and client facing data query tools. The data are comprised of images captured approximately every seven to ten metres from an array of optical cameras mounted vertically in four stereo pairs facing forwards, backwards, left and right, situated atop a mobile mapping vehicle. Figure 3-1 shows the proprietary Earthmine Mars imaging system mounted on the roof of a vehicle.

These images are processed in a bulk operation to generate location information for the image points via stereoscopy; the photogrammetric process of inferring depth from binocularly aligned images. Further proprietary post-processing is conducted to clean up the location estimates. An example of the resulting panorama can be seen in figure 3-2 where the viewpoint is located ahead and to the left of the mapping vehicle, looking back towards the mapping vehicle. The black patch on the road in the centre of the image shows the location of the mapping vehicle. The black regions in general are those parts of the scene where not enough reliable sensor information is available to the stereophotogrammetric process to reliably infer depth information. This typically includes occluded regions (the underneath of the mapping vehicle itself), regions that are too distant, and regions that present too small or thin a target area for the cameras given the required resolution for resolving points stereographically (*e.g.* the thin pole of the road light in the background is not produced even though the laterally

¹<http://www.earthmine.com>

²https://en.m.wikipedia.org/wiki/HERE_Map



Figure 3-1: The Earthmine Mars Collection System collects high resolution stereo panoramic imagery. Copyright Earthmine Australia 2015.

protruding lights are). The processed points are seen radiating outwards from the location of the mapping vehicle.

The post-processing phase generates rectified square images of the front, left, back and right viewpoints from the mapping vehicle, each viewpoint having a 90° field of view, and these images are aligned with the ground plane. The raw data (as seen in figure 3-2) is not made directly available to the client, but each pixel in these images can be queried using Earthmine's data access server (hosted either locally or remotely) to retrieve the associated location given as latitude, longitude and altitude values. By retrieving location information for each of the image pixels in the four viewpoint images, corresponding depth maps can be generated. Each pixel's latitude, longitude and altitude is converted into a Euclidean coordinate system by mapping the location parameters through the WGS84 ellipsoid. By setting the origin of these points to be the latitude, longitude and altitude of the imaging system, the depth z to any point \mathbf{p} in the images is calculated simply as

$$z = \mathbf{f}\mathbf{p} \tag{3.1}$$

where \mathbf{f} is the normalised focal vector of the image plane and points into the scene. Since the field of view is never greater than 90° , the depth value z is always non-negative.

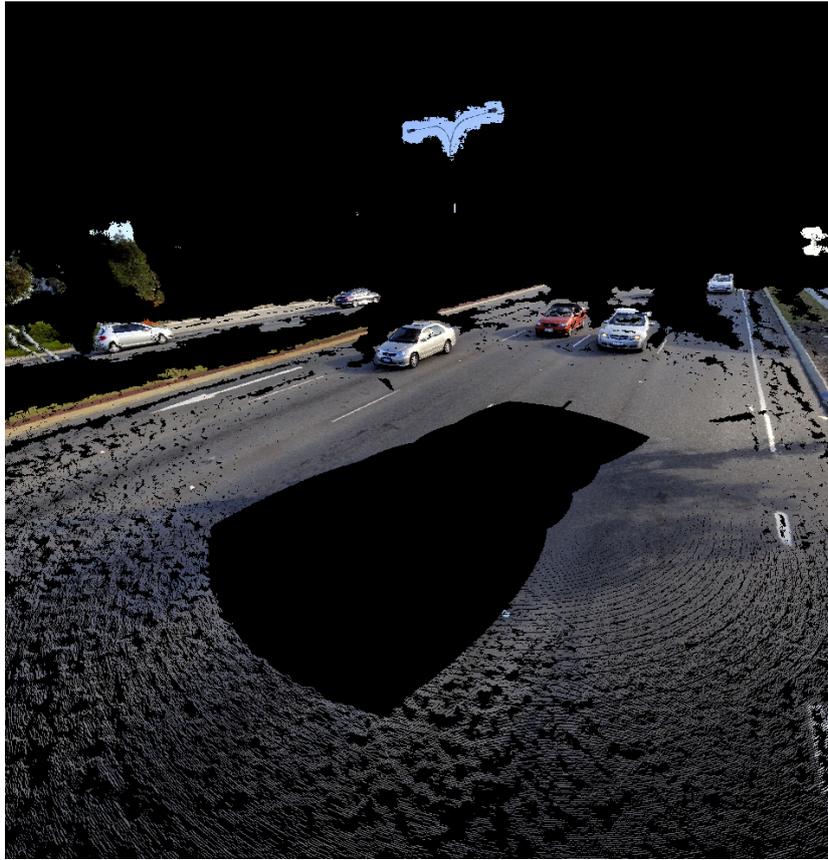


Figure 3-2: Visualising an example point cloud taken from the Earthmine dataset within the custom point-cloud viewer application.

Stereophotogrammetry is generally not a very accurate method of deriving depth information from a scene (compared with direct measurement techniques such as laser scanning). As such, the generated depth maps provide only coarse estimates of object locations, and only then when the objects are of sufficient size. The process also lacks accuracy in being able to determine clean boundaries in depth; the location data are erroneous, and the severity of these errors increases quickly with distance. There are also issues with the incorrect interpolation of depth information, leading to erroneous jumps in depth where none exist, or missing depth information entirely. Figure 3-3 shows the rear view from the mapping vehicle of a typical scene in the dataset, with the left image showing the RGB colour image and the right view showing the corresponding depth map. Lighter shades of grey in the depth map represent closer points, and the image has been contrast enhanced to better display the changing depth values. Black regions denote areas where no depth information is present (*e.g.* the sky) and these regions are masked out on the colour image. The depth map on the right is extremely noisy and it largely fails to accurately describe the boundaries of the objects present in the scene on the left, let alone the surface morphology of objects such as cars and street side

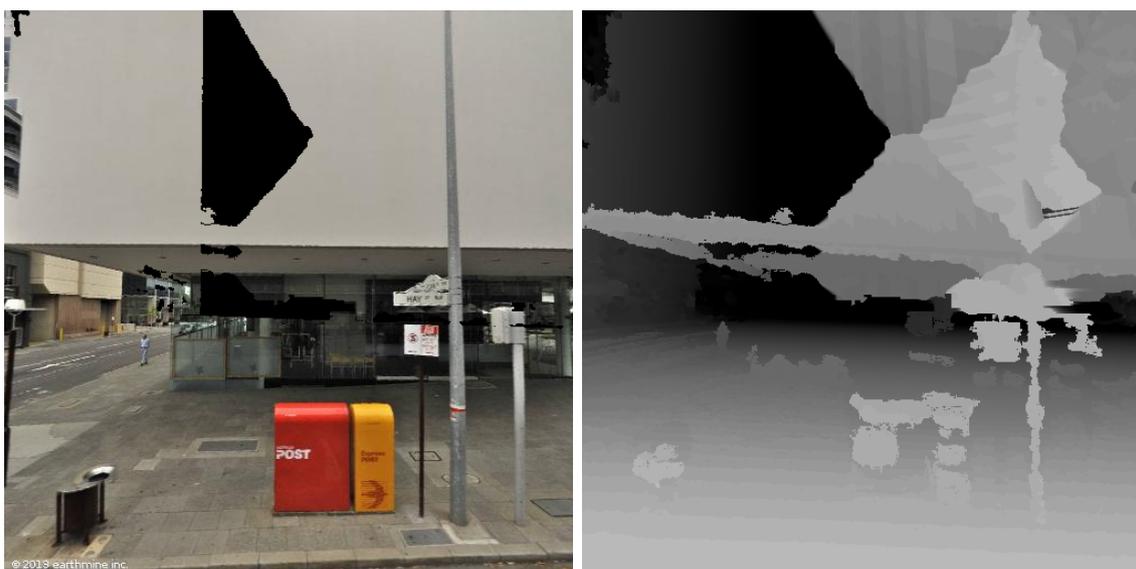
furniture.



Figure 3-3: An example scene taken from the Earthmine dataset showing the quality of the raw image data.

Figure 3-4 shows another example from the dataset. A post-processing step is used to interpolate depth values but this often results in poor quality depth information. In this example in particular, depth information is incorrect and missing from the building awning. In other locations such as around the postbox, garbage bin and street signs, depth information is only sufficient to indicate the presence of an object and its relative height above the ground. The thin posts of the parking signs are missing entirely showing that stereoscopy is poor in ascertaining detailed depth information. There are also issues with the colour images caused by incorrect stitching (visible in this example where the rim and opening of the garbage bin in the bottom left of the image is not properly centred).

The Earthmine dataset used in this work consists of a large part of the central business district of Perth, Western Australia. Stereophotogrammetry lacks the desired level of accuracy in being able to provide accurate geospatial positioning. For a long time, the greater costs associated with mobile mapping using laser scanners hindered its wider utilisation compared to cheaper stereoscopic approaches. However, laser scanning costs are now falling, driving growth in its popularity – particularly in mobile mapping. For these reasons, Earthmine has now abandoned purely stereophotogrammetric methods as its primary means of generating point cloud data, and now combines photographic imagery with laser scanners as part of the



(a) Rectified colour image.

(b) Corresponding depth map.

Figure 3-4: An example of incorrect depth interpolation in the Earthmine dataset.

HERE mapping company³. Due to the availability of laser scanned point clouds data, as well as its improved precision over stereoscopic methods of 3-D point cloud generation, a second dataset produced using laser scanning was sought.

3.3 The AAM Dataset

AAM⁴ specialise in the collection, processing, analysis and delivery of geospatial information including data captured via fixed terrestrial, and mobile (terrestrial and airborne) laser scanners. The dataset used in this work was generated using a single headed RIEGL laser scanner – the 3DLM StreetMapper (3D Laser Mapping, 2015). This laser profiling system is capable of collecting detailed spatial information with a full 360° field of view, a 300 metre range, and a sensor capacity of 300,000 measurements per second. The laser scanner is combined with a Ladybug3 (Point Grey Research, Inc, 2015) 360° spherical imaging capture system mounted on the roof of a mobile mapping vehicle. The system is able to capture data at speeds of up to 120km per hour.

The laser profiling system uses mirrors that rotate through a full 360° to direct the lasers at

³<https://company.here.com/here/>

⁴<http://www.aamgroup.com>

a planar angle that is lateral to the mapping vehicle’s direction of travel; data are captured in tiled cross sections as the mapping vehicle travels up the road. For each laser pulse, the mirror’s angle is recorded and time of flight measurements taken so that XYZ coordinates of the scanned objects within the laser scanner’s reference frame can be calculated. At each capture location, a minimum of six passes are made at a 500 kHz capture rate. The vehicle’s on-board Global Navigation Satellite System (GNSS) provides positioning information which is used together with orientation information recorded at each imaging position to calculate dense point clouds around the mapping vehicle with accuracy of approximately 2 centimetres out to a range of 150 metres.

The Ladybug3 spherical imaging system captures colour information about the scene, and the two datasets are combined to generate colourised 3-D point clouds centred on the mapping vehicle. Finally, after further post-processing, the data are turned into point cloud “strips” for saving into individual files in LAS(er)⁵ format; each file containing in the order of five million points. An example of one of these point cloud strips (viewed from above at an oblique angle) is shown in figure 3-5.

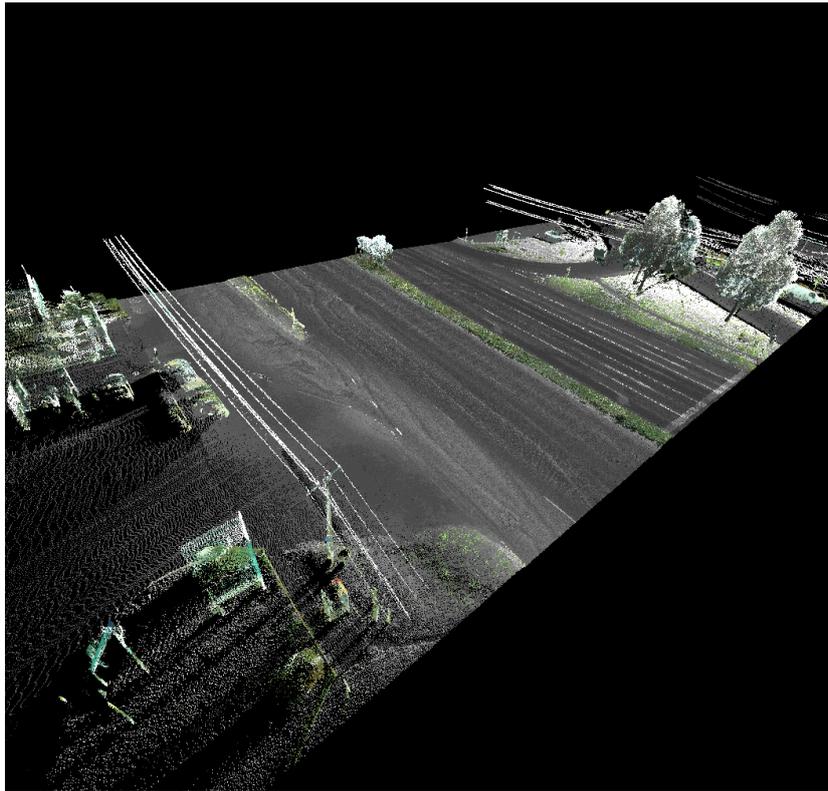


Figure 3-5: An example point cloud in the AAM dataset.

⁵<http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html>

The data used in this work are collected from along a seven kilometre stretch of Geelong Road in Footscray, Victoria, Australia. In order to produce the required 2-D images from these 3-D point cloud strips, a custom viewer was built to view the point cloud data from the point of view of the mapping vehicle (using GNSS data recorded within associated metadata files). The application is used to generate perspective projections of the point cloud data from the front, left, back and right of the mapping vehicle. From these views, both depth and colour images are saved as standard PNG format images (PNG is used over JPEG to avoid lossy compression artefacts). A non-standard integer based format is used to retain the floating point accuracy of the range values to millimetre precision out to 100 metres. Generating the images in this way allows the quality of the depth information in the AAM dataset (generated via laser scanning) to be compared against the stereoscopically produced Earthmine depth data. Figure 3-6 shows the colour and depth images generated from an example scene in the dataset. Point cloud resolution decreases with distance, so small holes in the depth data are interpolated however larger gaps are left as they are to avoid introducing spurious information. The data are supplied as point clouds rather than images (as in the case of Earthmine), so the generated colour images represent perspective projections of the points from the requisite viewpoints, hence the quality of the colour imagery is poor in comparison to the Earthmine colour images. As with Earthmine, areas outside of the prescribed depth range of 100 metres are masked out since they are too distant to be useful for object recognition.

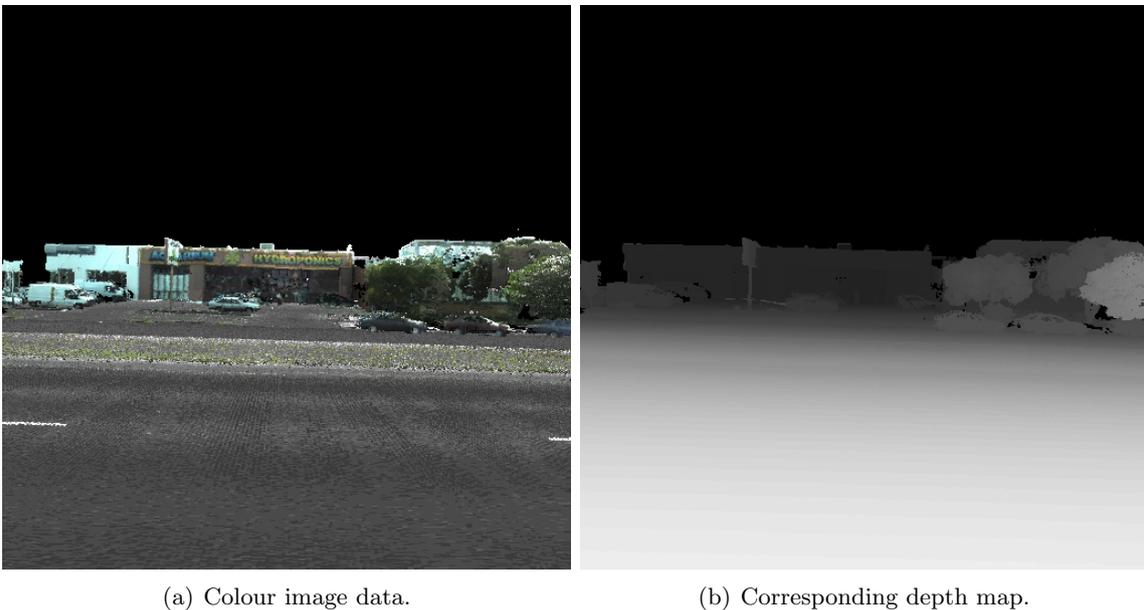


Figure 3-6: An example of scene from the AAM dataset (looking left from the mapping vehicle).

Even though laser scanning instead of stereoscopy is used to produce the AAM data, it has

issues of its own. Firstly, the colour images generated from the point cloud data is of very poor quality. The mapping of pixels from the images captured by the Ladybug imaging system to the spatial point data generated by the laser scanners is imperfect. This can result in colour registration errors such as the power lines shown in figure 3-5 being shown in white. Secondly, unlike the Earthmine dataset, the AAM dataset is much more limited in its diversity of scenery and object types since the area scanned is not within a densely populated region. The roads along which capturing was conducted are also generally very wide meaning that most of the interesting features are more distant than in the Earthmine data. In addition, the division of the data into strips that are orthogonal to the travel direction of the scanning vehicle means that interesting image content is nearly always only present in the generated left and right views. Finally, although the accuracy of the point cloud data (and thus the depth data) is much better than in the Earthmine case due to the use of laser scanning, objects that are closer to the mapping vehicle suffer from a “stippling” effect because different laser scan passes cannot be combined at such a close range. This results in some semi-transparent objects. However, given that most of the objects are more distant in the images, this does not present too great a problem. Many of these problems can be seen in figure 3-7.

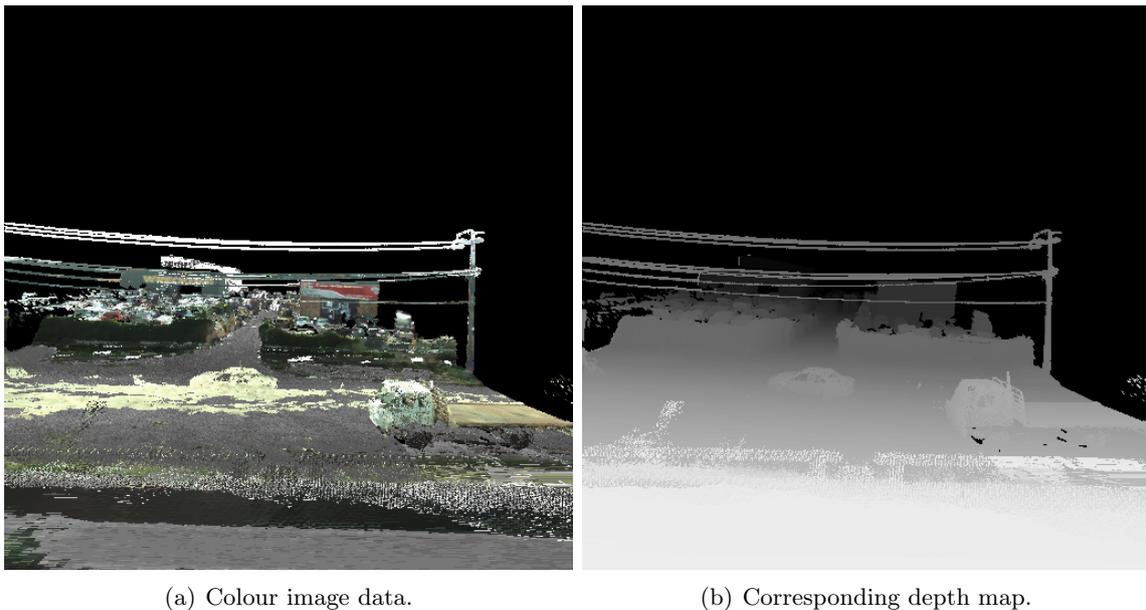


Figure 3-7: Problems with the AAM dataset including incorrect colour mapping and “stippling” of objects near to the scanning vehicle caused by the inability to register multiple laser scans at close range.

Neither the AAM or the Earthmine datasets perfectly capture 2-D colour images together with coregistered depth information. Although the AAM dataset is lacking in being able to provide accurate colour information, given the focus in this work to develop object recognition

techniques enhanced by depth information about the 2-D projection of a scene, it was decided to use the AAM dataset as the primary dataset since the depth information produced from laser scanning is both more accurate and less error prone than the depth data produced using the stereoscopic approach of Earthmine.

3.4 Ground Truthing

The object recognition methods developed in this work primarily involve supervised machine learning techniques. These methods entail providing an algorithm with many hundreds or thousands of examples of the kind of object that is to be searched for so that a generalisable model of the object can be statistically “learned” for later comparison against locations in query images. Comparisons against these models at particular locations in the query images indicate with a degree of confidence the presence of the objects of interest. These object hypotheses can then be validated and properly localised within the image.

Although supervised machine learning methods can give good object detection accuracy, providing sufficient data to create statistically accurate object models is a manually labour intensive and repetitive task; human volunteers must manually parse large image datasets to locate and label objects of the required type. This process of manually generating datasets of pre-labelled object examples is termed *ground-truthing*.

In order to use the AAM and Earthmine data for the purposes of object recognition, it was necessary to undertake a ground-truthing exercise on these datasets. This necessitated the development of an application (and ancillary software libraries) to support this task. For the ground-truthing exercise itself, participants were sought for their capability and willingness to focus for lengthy periods while identifying and tagging hundreds of examples of the required objects.

3.4.1 Application and Workflow Design

Two adult participants having Autism Spectrum Disorder (ASD) from Curtin University’s Department of Occupational Therapy were chosen to undertake the ground-truthing exercise because of their expected ability to perform at consistent work rates and levels of accuracy. The application user interface and the workflow of the task itself was specifically designed to be straightforward and to cater to the needs and abilities of the participants so as to facilitate their working efficiency in this task, while minimising the need for supervision.

The application was initially produced as a way of visualising images from the dataset and for selecting and exporting arbitrary regions of the data to image files. For the purpose of the ground-truthing exercise, a number of significant modifications were made to incorporate the addition of object type labels and recording of instances into a file store.

An initial version of the application along with a description of the task and the expectations for how the work would be undertaken was presented to the participants, and the participants were encouraged to give feedback on how they thought the exercise would go. As a result of these discussions, further modifications were made to address several issues that were identified with the application and the expected workflow. Firstly, the ability to navigate and visualise the point-cloud data was removed as it was felt that this feature would be too distracting while not sufficiently enhancing the users ability to discern the presence of objects in the data. Secondly, due to concerns over the finality of object extraction decisions, an option was incorporated to remove an already extracted object from the ground-truthed set (prior to this, the option was only available by manually editing the ground-truth store). Thirdly, a set of keyboard strokes was incorporated to enhance the fine control required to accurately place a bounding box around an object. The W,S,A,D keys were chosen to allow the right-handed users to work with the system in a comfortable and familiar way. Finally, the two users decided to parse the data in filename order from either end of the data directory to avoid working on already processed images and potentially disagreeing about the extraction of particular objects.

Through this iterative process of design and evaluation, a tailored application and workflow was developed to maximise the ability of the users to undertake the ground-truthing exercise. This was confirmed to be the case from making random assessments of the quality of object extractions, and from tracking the work rates of the two users.

Figure 3-8 shows a screenshot of the application being used to determine the bounds of traffic lights in the left facing view of a capture location in the AAM dataset. The interface allows the user to interact with the Earthmine and AAM datasets in various ways. Individual panoramas are loaded using the “Open Panorama” button on the bottom right. A particular view direction is selected using the “Face” buttons in the bottom left. The type of view being displayed – either the colour image, or a contrast scaled representation of the depth map, is selected using the “View Type” buttons. New object types are defined with the “New Class” button which displays a dialogue asking the user for the label to be used to identify the new object category. The drop down list underneath this provides the currently defined list of object categories. The currently selected object type (shown in the figure as “traffic light”) is the label associated with new bounding rectangles drawn by the user in green. All of the ground-truthed extracts for the currently selected object type are shown with blue bounds.

The button labelled “Extract” fixes the association between the ground-truth bounding box drawn in green and the currently selected object type, changing the green rectangle into a fixed blue rectangle and recording the ground-truthed example in memory. The complete set of ground-truth examples are written to file in a standardised text format upon application exit.

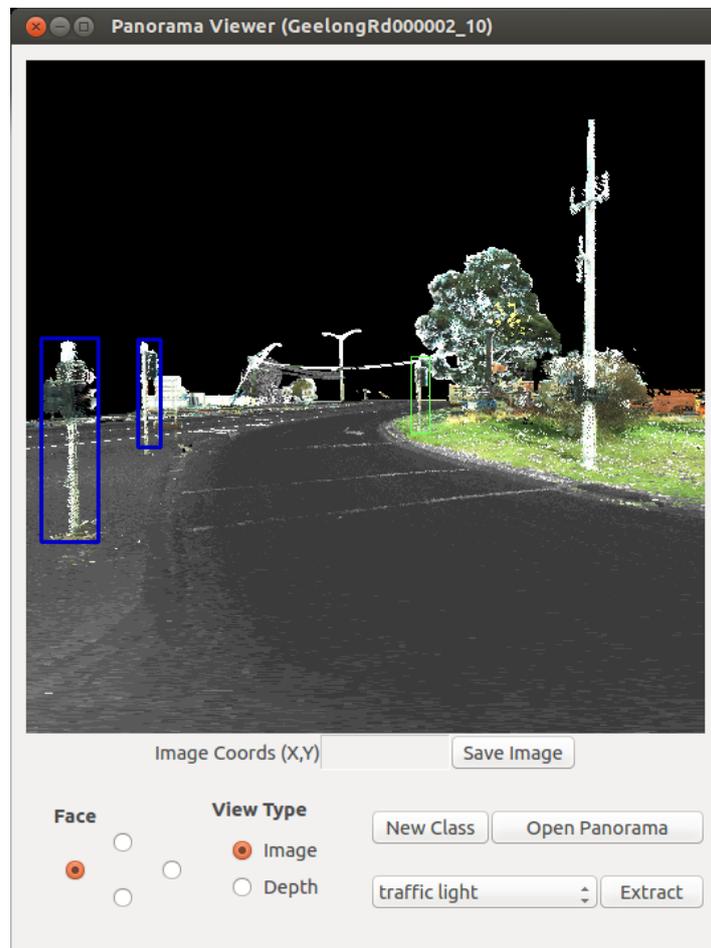


Figure 3-8: The user interface for the custom ground-truthing application.

3.4.2 Task Details

Prior to commencement of the ground-truthing exercise itself, each LAS file in the AAM dataset was processed to extract the colour and depth information for the four views (front, left, back and right) and these images were translated into a source agnostic file format readable by the ground-truthing application. For the ground-truthing exercise, each of these files was loaded in turn into the application whereupon the participant made a visual assessment

of the colour and depth images to identify the required objects of interest. A list of desired object types was defined beforehand based on a sample viewing of the data. Upon identification of an object, its type was selected from a drop down list, and a bounding box was positioned around the region of the image containing the object using the mouse. The user then pressed a button to save the meta data for the tagging of that object including the object’s type, the data file, the view within the file (front, left, back or right), and the location and dimensions of the bounding box set by the user.

A number of different object types were identified for ground-truthing. Table 3.1 shows the object types identified from the AAM dataset along with the number of objects eventually extracted (after post-processing).

Object	Count
Bus	12
Bus Shelter	24
Bus Stop Post	22
Car	729
Garbage Bin	16
Other Road Sign	1491
Parking Sign	8
Person	46
Rectangular Road Sign	668
Road Light	1461
Round Road Sign	23
Street Lamp	10
Telegraph Pole	937
Telephone Kiosk	20
Traffic Light	1172
Triangular Road Sign	163
Truck	104
Van	84
TOTAL	6990

Table 3.1: Object types and counts from the ground-truthing exercise on the AAM dataset.

The appearance of objects in the data varied significantly. In order to provide a reliable dataset for submission to the machine learning algorithms, the appearance of an object needed to meet certain criteria before it was deemed suitable for labelling and recording as part of the ground-truth dataset for that object type. These criteria were:

1. Objects on the list of candidate object types,
2. Objects wholly contained within the image, and

3. Objects not occluded by other objects (whether or not of the same type), or by other scene elements.

For certain object types, the last of these criteria needed to be loosened. For example, the presence of a small number of tree branches occluding a more obvious road light did not greatly corrupt the appearance of the road light itself. However, instances of road lights more densely occluded by branches and foliage were disregarded as such examples risked introducing too much noise into the ground-truth which might potentially decrease classification precision. It was also necessary to more concretely define the bounds of certain object types *e.g.* how much wiring to include at the top of telegraph poles.

The nature of each object type was discussed in detail with the ground-truthing participants, along with what characteristics made for “good” and “bad” examples for a given object type. To avoid deliberating for too long about whether or not to tag a particular example, a maximum time of approximately three minutes per object was observed by the participants. If it took longer than this to decide if the example was suitable, the user moved on to the next view or the next example without extracting the object for inclusion in the object dataset.

These rules, and the design of the ground-truthing application allowed the participants to carry out the task completely unaided. After commencing the exercise, on only a single occasion did one of the participants request further clarification concerning the suitability of an object type. After the end of the exercise, Awk scripts were run on the files containing the ground truth meta data to ensure that no two bounding boxes overlapped or described the same instance, and that each bounding box was completely contained within its view.

3.4.3 Task Evaluation

Data on work rates were collected from the participants at the end of each four hour shift. These data were recorded whenever an object was extracted and included the file identifier, the view from which the object was extracted (front, left, right, or rear), the type of the object recorded, a time-stamp, and the location and dimensions of the bounding box placed around the object. From these data for each shift, the average number of objects per file (object density), and the average number of minutes spent on an object’s extraction (mins / object) were calculated. Tables 3.2 and 3.3 show these data for the two participants, and figure 3-9 shows plots of these data over the duration of the exercise. Participant 2 was able to maintain a consistent work rate over the duration of the exercise. Participant 1 was consistent in work

rate apart from a single data point indicating a slow down around the Christmas break. These charts also show no correlation between object density and work rate for either participant indicating that changes in the apparent complexity of an image (as measured by the object density) did not influence the speed at which the participants were able to identify and extract the objects in the images (although image complexity in the AAM dataset is qualitatively low). The data show that participant 2 was able to maintain a significantly higher work rate overall than participant 1 (nearly 2.5 times faster) and this effect is not explainable by the different average densities of objects seen in the panoramas by the two participants. The difference is best explained by the participants' differing capabilities.

Date	Obj. Density	Mins / Obj.
14/11	2.62	2.82
19/11	1.75	2.11
21/11	2.34	2.44
26/11	2.45	2.92
28/11	5.75	1.98
03/12	5.65	2.34
06/12	4.24	2.42
17/12	3.04	2.65
19/12	2.39	2.60
20/12	2.03	2.91
27/12	3.67	4.79
28/01	6.30	1.51
30/01	4.67	2.79
04/02	4.79	2.04
06/02	4.52	2.13
11/02	1.67	2.25
13/02	1.80	2.74
18/02	2.23	2.35
20/02	2.07	2.44

Table 3.2: Participant 1 work rates

Date	Obj. Density	Mins / Obj.
07/11	3.09	0.94
11/11	5.41	1.10
14/11	6.25	0.83
18/11	6.41	1.13
21/11	5.97	1.01
02/12	5.69	1.01
03/12	5.95	0.87
04/12	7.00	0.99
05/12	7.00	1.15
06/12	6.98	0.81
12/12	6.96	1.21
16/12	7.00	1.22
13/01	7.00	1.19
14/01	4.00	1.38
15/01	2.41	1.71
16/01	2.40	1.02
17/01	2.16	1.53
20/01	3.67	1.29
23/01	2.37	1.80
28/01	3.81	1.29
30/01	4.04	0.87
03/02	3.42	0.81

Table 3.3: Participant 2 work rates

The work rates of both users reached maximum efficiency within the first four hour shift and remained relatively consistent over the full duration of the task. In addition, a qualitative evaluation of a small set of examples from each shift provided confidence that the placements of the object extraction bounding boxes by both participants remained accurate over the duration of the exercise.

Although it was not possible to compare work rates and labelling accuracy against a baseline,

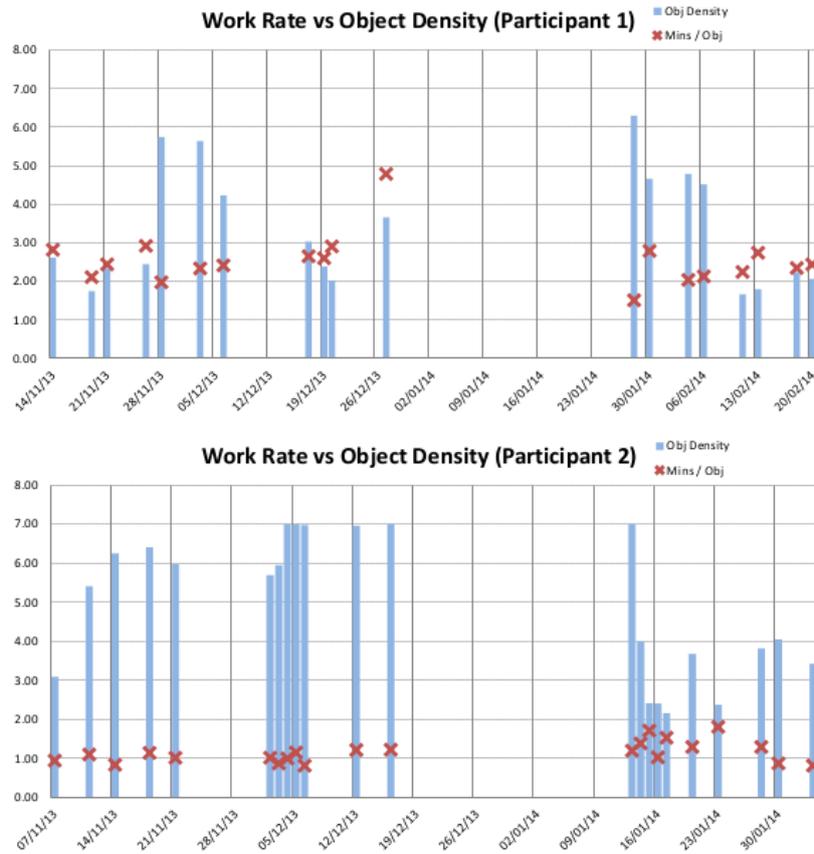


Figure 3-9: Participant work rates and object density

the ability of the users to maintain such relatively high and consistent work rates and quality on a very repetitive and mundane task, demonstrates the high level of competence that individuals with ASD can bring to such a task.

3.5 Evaluating Detection Accuracy

This section explains how recall and precision are calculated in this thesis to compare the accuracy of the different object classification and detection schemes being evaluated. These calculations differ from the previous standard methods of calculating object classification and detection metrics, and the reasoning behind these new metric derivations is justified in view of the limitations of the earlier methods that are unable to accurately represent both recall and precision within a standardised framework.

To understand the accuracy of a given object recognition scheme, data having known ground

truth labels are used. The aim is to compare the classification or detection score achieved by the recognition system, against these known labels. The two main performance metrics of interest are recall and precision, however a number of other statistics that assess the accuracy of an object classification / detection scheme can be calculated. All of these statistics are computed from four core statistics derived from the classifier’s ability to correctly label positive and negative examples – respectively, true-positives T_p , and true-negatives T_n – and its inability to correctly label positive and negative examples – respectively, false-negatives F_n , and false-positives F_p .

In object detection, a query image is parsed to discover if any objects of interest are present in the image, and where they are located. This might be achieved using a sliding window which iteratively selects subregions from the query image for classification. Classification can therefore be treated as a special case of detection where the problem is one of choosing whether a given image represents the object of interest or not; either the whole of the image is considered as the object or none of it is. Detection allows for situations where a given subregion of the query image encompasses only a portion of an object. In both cases, the classification response to whether the image extract is or is not an example of the object of interest is given by a confidence value ψ denoting how well the extract fits the classifier’s pre-trained model of the object. The range of this value is classifier dependent but it can be interpreted as a probability that the extract is an example of the positive class by standardising it between 0 and 1. With an unbiased confidence threshold, values of $\psi \geq 0.5$ indicate classification as a positive example and values of $\psi < 0.5$ indicate classification as a negative example. However, since the classifier itself may be biased, these confidence values are stored so that metrics can be calculated at different thresholds once all of the responses have been collected.

3.5.1 Evaluating Object Classification Accuracy

For object classification, two sets P and N are maintained to store the response probabilities for the classification of the positive and negative validation examples respectively. After each classification, in the case of positive examples, ψ is added to set P . In the case of a negative example, $1 - \psi$ is added to set N . Once all classification responses have been collated, for a given value of the threshold $t \in [0, 1]$, let

$$P_t = \{\psi \mid \psi \in P, \psi \geq t\} \tag{3.2}$$

and let

$$N_t = \{\psi \mid \psi \in N, \psi \geq t\}. \tag{3.3}$$

The core statistics can then be calculated as

$$T_p = |P_t| \quad (3.4)$$

$$F_n = |P| - T_p \quad (3.5)$$

$$T_n = |N_t| \quad (3.6)$$

$$F_p = |N| - T_n \quad (3.7)$$

From these core statistics calculated at the given threshold, the summary statistics used for evaluating the accuracy of the classifier can be derived. The *Positive Predictive Value* (PPV) (also called *precision*) specifies the ratio of correct classifications to all positive classifications. The *True Positive Rate* (TPR) (also called *recall*) specifies the ratio of correct classifications to all the actual positive examples tested. The *False Positive Rate* (FPR) (also called *fallout*), gives the rate at which the classifier spuriously classifies negative data as positives. The overall accuracy (ACC) specifies the ratio of correctly classified examples (positives plus negatives), however this will give a biased result if the number of positive and negative examples do not match. The F1 statistic is calculated as the harmonic mean of precision and recall and it can be useful in situations where a single metric that encompasses information about precision and recall is preferred. The harmonic mean is less biased toward large discrepancies between the precision and recall values.

$$PPV = \frac{T_p}{T_p + F_p} \quad (3.8)$$

$$TPR = \frac{T_p}{T_p + F_n} \quad (3.9)$$

$$FPR = \frac{F_p}{F_p + T_n} \quad (3.10)$$

$$ACC = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \quad (3.11)$$

$$F1 = 2 \left(\frac{PPV \cdot TPR}{PPV + TPR} \right) \quad (3.12)$$

By varying t smoothly from 0 to 1 and calculating the core statistics at each value of t , the behaviour of the classifier at different confidence levels can be evaluated. This is important because even if a large number of training data are provided to the classifier, depending on the features used, it may estimate a classification boundary that is either too closely fitting to the training data (possibly resulting in a degradation in recall), or too loosely fitting (possibly resulting in a degradation in precision). Of greater importance is the relationship between precision and recall over a range of thresholds since this indicates how well the classifier is able

to discriminate between the two classes. These changing values of precision and recall with threshold are usually shown graphically with recall along the X axis and precision along the Y axis. Each value of $t \in [0, 1]$ defines a point in this plane, points which when taken together define a curve. The closer the area under this curve is to 1, the better the discriminative ability of the classifier. However, the classifier may be biased towards either better recall or precision, and for a given object recognition task, one of these metrics may be more important and so the area under the curve is usually too simplistic a measure to use to characterise the behaviour of the classification scheme.

Examples of precision-recall curves are shown in figure 3-10 (this example is taken from section 4.3.9). The graph shows different curves representing the accuracy of different feature extractors on a single dataset.

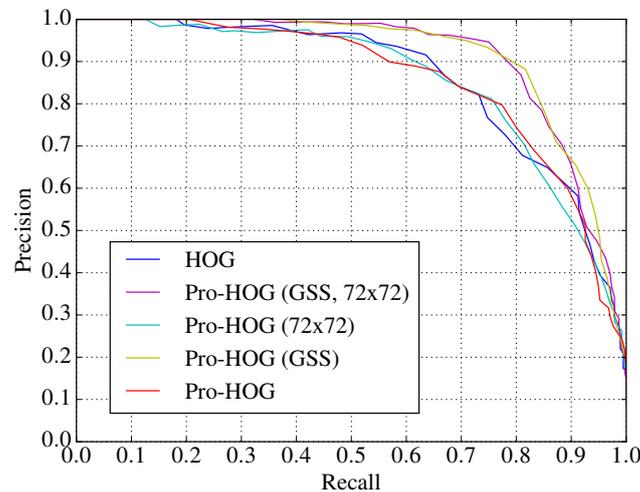


Figure 3-10: An example of precision-recall curves (extracted from section 4.3.9)

The behaviour of the precision and recall metrics can behave quite differently from one another given changing feature extraction parameters. For this reason, some of the graphs in this thesis (in particular, those throughout section 4.2) show recall and precision separately as the dependent variable at a fixed classification threshold (nominally, the default) as the parameter under evaluation is changed.

3.5.2 Evaluating Object Detection Accuracy

For object detection, the calculations of the core statistics differ since the degree to which the detection regions overlap with the ground truth rectangles is important. For the two class detection problem, a validation set of images $I \in Q$ with ground truth rectangles uniquely identifiable to each image G_I are defined. Let $G \supset G_I$ with the imposition that

$$\forall g, h \in G_I \mid \text{area}(g \cap h) = 0$$

where $\text{area}(x)$ is defined as the number of pixels in region x .

The images are presented to the object recognition scheme one at a time, and a scanning algorithm classifies successive rectangular subregions $r \in I$ (with r uniquely addressable to I). Each response ψ giving the confidence of the classification of subregion r as a positive example is recorded in a detection set D as a 2-tuple $\mathbf{d} = \langle r, \psi \rangle$ under the condition that

$$\forall \mathbf{b}, \mathbf{d} \in \{ \langle r, \psi \rangle \mid \langle r, \psi \rangle \in D, \langle r \in I, \psi \rangle \} \mid \text{area}(\mathbf{b}_r \cap \mathbf{d}_r) = 0.$$

After scanning all images in Q , for detection confidence thresholds $t \in [0, 1]$, let

$$D_t = \{ \langle r, \psi \rangle \mid \langle r, \psi \rangle \in D, \langle r, \psi \geq t \rangle \}.$$

The values of the core statistics can then be calculated as

$$T_p = \sum_{g \in G} \sum_{\mathbf{d} \in D_t} \text{area}(g \cap \mathbf{d}_r) \quad (3.13)$$

$$F_n = \sum_{g \in G} \text{area}(g) - T_p \quad (3.14)$$

$$F_p = \sum_{\mathbf{d} \in D_t} \text{area}(\mathbf{d}_r) - T_p \quad (3.15)$$

$$T_n = \sum_{I \in Q} \text{area}(I) - T_p - F_n - F_p \quad (3.16)$$

The core statistics are given in terms of queried pixels but they can be normalised with respect to the total area of the images in the query set if desired without affecting the derivation of the summary statistics in equations 3.8—3.12. Importantly, this means that the same mathematical framework for deriving these statistics can be used when evaluating object classification accuracy, and when evaluating the accuracy of localising detected objects in query images.

Figure 3-11 shows a positive ground truth bounding box (blue outline) located somewhere

within a query image superimposed by a subregion of the query image being classified (red outline). Every part of the query image not within ground truth bounding boxes is considered as negative data. The text labels indicate which parts of the image account for the calculation of the different core statistics.

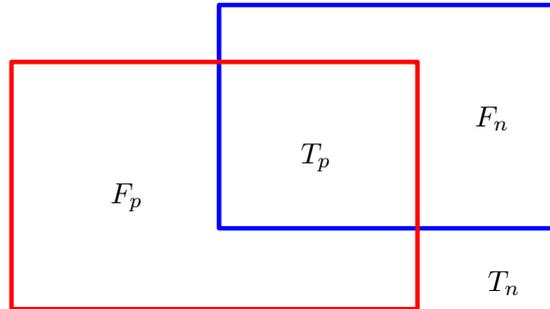


Figure 3-11: A dense sampling of image subregions is classified. Areas of the image are used to calculate the four core statistics according to how the detection subregions (red boxes) overlap the ground truth bounding boxes (blue boxes).

This method of calculating the core statistics is preferred in this work for evaluating the accuracy of object detection because it allows for detections that align more closely with the ground truth labelling to contribute to heightened measures of accuracy (primarily precision and recall). For a given labelling of ground truth instances in an image (for example, as might be labelled according to the manual placement of rectangular bounding boxes), an object detector that is deemed to be more accurate should be able to more closely match its detections to those labellings.

This framework for deriving object detection accuracy metrics is also extensible to ground-truth and detection regions having non-rectangular bounds. While the calculations are simpler using rectangular regions, because the definition of the area function relies only upon the number of bounded pixels, arbitrarily shaped regions can be used. This allows for the evaluation of object detection schemes where ground-truth objects are given using exact segmentations. Due to the added complexity of producing such exact ground-truthed segmentations, the ground-truthing exercise described in section 3.4 required only that bounding rectangles around the object examples be defined.

3.5.3 Problems with Alternative Evaluation Criteria

The PASCAL Visual Object Classes (VOC) Challenge (Everingham *et al.*, 2015) is a 2-D object recognition challenge that ran between 2005 and 2012. One of the aims of this challenge

was to stimulate interest into researching difficult object recognition tasks. During the period in which the challenge ran, much effort was dedicated by many researchers towards the development of object classification and detection schemes that could score well according to the detection evaluation criteria stipulated by the challenge. The criteria used for the challenge counted a ground truth object as detected if the intersection of a detection rectangle with a ground truth rectangle was greater than 50% of the area given by the union of these regions

$$a_o = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \quad (3.17)$$

where $a_o > 0.5$ denotes the detection of the instance bounded by rectangle B_{gt} according to detection bounding box B_p (Everingham *et al.*, 2015). The reason given for setting the detection overlap threshold at 50% is that the rectangular ground truth labelling of real world objects necessarily involves a degree of subjectivity, and detection schemes should not be penalised for having a slightly imprecise interpretation of the object’s placement in the image. However, this justification does not account for two observations. Firstly, although the placement of a ground truth bounding box for a single instance may suffer from a degree of subjectivity, there ought not to be any systematic (non-random) bias in the placement of the bounding boxes in relation to the “correct” locations of the objects in the images, but if the bias that exists *is* systematic, then it should in fact be interpreted as the correct labelling of the object; detectors should still seek to match these ground truth bounding boxes as closely as possible. Further, any bias in the placement of the ground truth rectangles present in the data should be reflected equally in the portions of the dataset used for training and for validation. If systematic bias is only present in either the training data or the validation data, then this is a problem with the ground truth labelling of the data.

Secondly, although a given object detector may be penalised for imperfectly matching detections, given enough test data, a better object detector should still be able to match its detections to the known ground truth labels more accurately *on average*. The detection criteria given in Everingham *et al.* (2015) cannot assist in evaluating in sufficient detail the accuracy of detections – especially in the case of making a comparative analysis of the accuracy of two or more detectors. If two different object recognition schemes both manage to correctly detect the same number of ground truth objects, but the first detector can more accurately ascertain the positions of these objects, then the evaluation criteria given by Everingham *et al.* (2015) cannot differentiate between the schemes. While it is true that applying segmentation to the results of detection from the two recognition schemes might ultimately yield equivalent results, this assumes that segmentation is not already used as part of the detection process for the schemes. In any case, a less accurate initial detection of an object will be less likely to subsequently allow for a perfect segmentation of that object.

The use of an overlap threshold to count detections in an “all or nothing” sense, can give the false impression that an object detection scheme performs very much more accurately than it does in reality. Defining rectangular bounding boxes is also not the only way of labelling ground truth data (although this is arguably the easiest, and tends to be the most commonly used method). Both detection regions and ground truth regions can be defined using non-rectangular regions the area of a region can be defined simply as a pixel count. Since the new evaluation criteria described herein measures accuracy in terms of pixel counts, it is more generally applicable than methods of evaluation that count arbitrarily sized rectangular regions. Assuming a labelling of the ground truth is available at a pixel level, calculation of these evaluation criteria can even be undertaken after further validation such as object segmentation has been carried out. The statistics as newly derived herein are then not restricted to only evaluating the results of object detection; they may also be used to evaluate the accuracy of the output from a complete object recognition scheme.

Deriving the core statistics for object detection by reference to the number of detected pixels, can also be applied to object classification instead of using equations 3.4—3.7 which can be shown to be equivalent to those of equations 3.13—3.16 given the requirements of classification. In object classification, the detection rectangles are the same as the ground truth rectangles (the classifier is presented with a validation set which is comprised of the ground truth positive and negative examples themselves), and the dimensions of the rectangles bounding each example are no longer important and may be set to an area of one pixel. The rectangle locations are irrelevant. The classification conditions thus allow for the definition of the positive ground truth extracts $G = \{r \mid area(r) = 1\}$, where r is a uniquely identifiable extract, and $D = G$. Letting $D_t = \{\langle r, \psi \rangle \mid \langle r, \psi \rangle \in D, \langle r, \psi \rangle \geq t\}$ for confidence threshold $t \in [0, 1]$, equation 3.13 then becomes

$$T_p = \sum_{g \in G} \sum_{d \in D_t} area(g \cap d_r) = \sum_{d \in D_t} area(d_r) = |D_t|. \quad (3.18)$$

D_t simplifies to the definition of P_t (equation 3.2) because the locations of r in D_t are irrelevant and the area of every example is equivalent. This means that

$$T_p = |D_t| = |P_t| \quad (3.19)$$

which is the same as equation 3.4. By this rationale, the method of deriving the evaluating criteria as given in equations 3.13—3.16 can be used for evaluating the results of object classification, detection, and segmentation (given pixel resolution ground truth labelling).

Chapter 4

Scale Proportionate Histograms of Oriented Gradients

Scale Proportionate Histograms of Oriented Gradients (Pro-HOG) is a feature extraction technique based on the Histograms of Oriented Gradients (HOG) feature extractor of Dalal and Triggs (2005). Pro-HOG encodes contrast gradient characteristics about local image regions at a predefined fixed descriptor resolution without first resizing the input image extracts. These descriptors can then be used to assist in the classification of image extracts.

Defining the resolution of the features encoded by the descriptor, offers two possible advantages over methods that first undertake image resizing before computing descriptors over the image. Firstly, the act of resizing an image extract can change the quality of information it contains. Scaling an image down loses information because fewer digital elements (pixels) are available to encode the information. Scaling an image up requires interpolating data between pixels which may introduce spurious information (depending on the effectiveness of the scaling algorithm used which the feature extractor normally does not explicitly implement as part of its algorithm). Pro-HOG seeks to avoid either of these two situations to maintain an accurate feature encoding generated from the extract at its original pixel resolution.

Secondly, the need to resize image extracts prior to the generation and comparison of descriptors is inefficient when a very large number of descriptors must be generated. This is typically the situation in object detection where the task of identifying the objects of interest in a query image necessarily involves the creation of very many descriptors from many different, sometimes overlapping, image locations. By avoiding the need to resize image extracts, Pro-HOG offers a much more efficient and accurate method to extract features.

This chapter describes the approach, design and implementation, and empirical evaluation

of Pro-HOG compared to the classic HOG feature extractor described in section 4.1. In section 4.2, the different parameters of Pro-HOG are evaluated in terms of their impact on classification accuracy in a binary object classification task. In section 4.3 classification experiments are carried out over 15 different object types from three different datasets and using two different types of input data – grey scale/intensity images derived from colour images, and for two of the datasets, depth imagery acquired using laser scanning or stereoscopy. The findings from these experiments are discussed in section 4.4 and follow up experiments are carried out in lieu of these findings to answer questions arising from the experiments in the previous section. The chapter concludes with section 4.5 which summaries the findings regarding Pro-HOG’s utility for object classification versus HOG. It is concluded that Pro-HOG is just as capable as the original HOG algorithm in being able to accurately encode objects for the purposes of classification, and may in some circumstances (*e.g.* for smaller objects) allow for increased classification accuracy – especially in recall.

4.1 Pro-HOG Design

This section describes the approach and implementation of Pro-HOG and its differences with the classic HOG extractor of Dalal and Triggs (2005) upon which it is based.

4.1.1 Histograms of Oriented Gradients

Histograms of Oriented Gradients (HOG) (Dalal and Triggs, 2005) are a very popular and well performing type of feature. The HOG feature extractor encodes contrast orientation and magnitude over neighbouring local areas of an image. Contrast, being the first derivative of image intensity, is especially good at encoding image structure especially at the boundaries of objects because it is more tolerant to variance in illumination.

HOG feature extraction proceeds in the following manner: An image is first resized to predefined pixel dimensions (according to a given parameterisation of the algorithm) and this is then divided up into a grid of equivalently sized, nominally square, cells. Over each of the cell subregions, the contrast is sampled and encoded in a histogram of predefined length. Finally, each cell’s histogram is concatenated with three of its neighbouring cell histograms into normalised 2×2 blocks. This results in a total feature vector length of $4N(\mu - 1)(\nu - 1)$ where N is the histogram length, and μ and ν are the width and height dimensions of the grid of cells.

HOG feature vectors are used in this chapter at the global level (over the whole of an extract that bounds an object of interest). Dividing the extraction area into square cells comprised of many pixels means that the descriptors express localised information as well as spatial relationships between these cellular regions because the cells are grouped together into overlapping blocks from which contrast information is summarised into the fixed length histograms. In this way, they encode both global and local information about an image extract.

HOG features have been used widely in a number of object recognition schemes and have demonstrated usefulness over a large variety of different object and image types (Dalal and Triggs, 2005; Felzenszwalb *et al.*, 2010; Schroff *et al.*, 2008; Gao *et al.*, 2011). HOG gives good general performance and its underlying method of extracting contrast gradients is widely used as the basis for other feature extractors and interest point detectors. As such, it is used here as the basis for Pro-HOG.

The core difference between Pro-HOG and HOG is that fixed feature vector lengths are managed using *integral images* (Crow, 1984) instead of using images that have been resized to fixed pixel dimensions. Integral images allow for the efficient computation of the sum of any subregion of an image matrix by way of a simple arithmetic calculation over the four corners of the subregion. The contrast gradient magnitudes are computed once over all pixels, but unlike the integral histogram of Porikli (2005), the magnitudes are stored in integral images that correspond to the histogram bins given by the gradient orientations. Instead of propagating the construction of the histogram according to the changing location in the image (meaning that several integral histograms must be maintained if the image is to have features extracted from it at different scales, with the correct orientation bins being found for each extraction), arbitrary subregions of the image can be selected to extract the histogram data from in constant time.

Figure 4-1 shows an example of how a simple single valued image matrix (on the left) can be encoded as an integral image (on the right). The sum over the blue region in the matrix on the left can be calculated as $(A + B) - (C + D)$ where A and B are respectively the values at the “outer” top left and “inner” bottom right diagonals of the blue square, and C and D are respectively the values at the “outer” bottom left and the “outer” top right of the blue square. The “outer” square is shown in lighter blue in the right hand matrix in figure 4-1.

The contrast gradient orientations and magnitudes are calculated at the original pixel resolution of the image, so no resizing is conducted that could degrade the quality of any image structure and so the quality of the information encoded in the descriptors is maintained. Scale-space image pyramids are not required to store the image data at different sizes to ac-

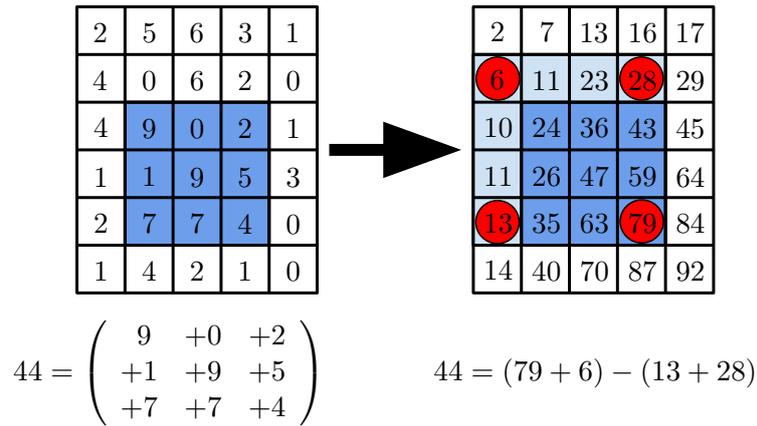


Figure 4-1: Conversion of an image matrix (left) to an integral image matrix (right). The integral image allows for sums over arbitrary regions of the image (dark blue region) to be calculated by indexing into the integral image only at the four corners of the subregion.

count for the possible presence of objects at different scales. This means that object detection using a sliding window based approach can be conducted efficiently since feature vectors of arbitrary size and location can be extracted in constant time with only a few arithmetic operations after the initial pixel level calculation and storing of the gradient magnitudes. Objects can be detected at arbitrarily fine scales instead of at the fixed scale intervals predetermined by the use of image pyramids because the dimensions of the image subregion over which features are to be extracted can be determined at run-time.

4.1.2 The Pro-HOG Algorithm

Feature extraction in Pro-HOG is performed in two phases. In the first phase, an image that encodes the gradient information at every pixel is produced. In the second phase, rectangular regions of arbitrary dimensions (up to the dimensions of the whole image) define the bounds over which the features are extracted. In the second phase, because the gradient image is held in memory as integral images and because the length of the feature vectors is fixed in advance, the final descriptor can be generated in constant time irrespective of the actual pixel dimensions of the rectangular subregions. This is especially useful during object detection, where the whole of the query image can be processed in the first phase, while the extraction of feature vectors from many candidate overlapping regions of the query image only entails the constant time second phase of processing. For object detection, this makes Pro-HOG very much more efficient than a naïve implementation of the HOG algorithm which redundantly processes pixel gradient orientations many times over for overlapping candidate detection regions. This feature of Pro-HOG makes it especially efficient for processing images in scale-space whether or not explicit depth information is available, but especially where it is not

and image-pyramids must be used, because overlapping regions must be evaluated to generate responses at the different scales.

Figure 4-2 gives a high level diagrammatic representation of the algorithm. The diagram shows

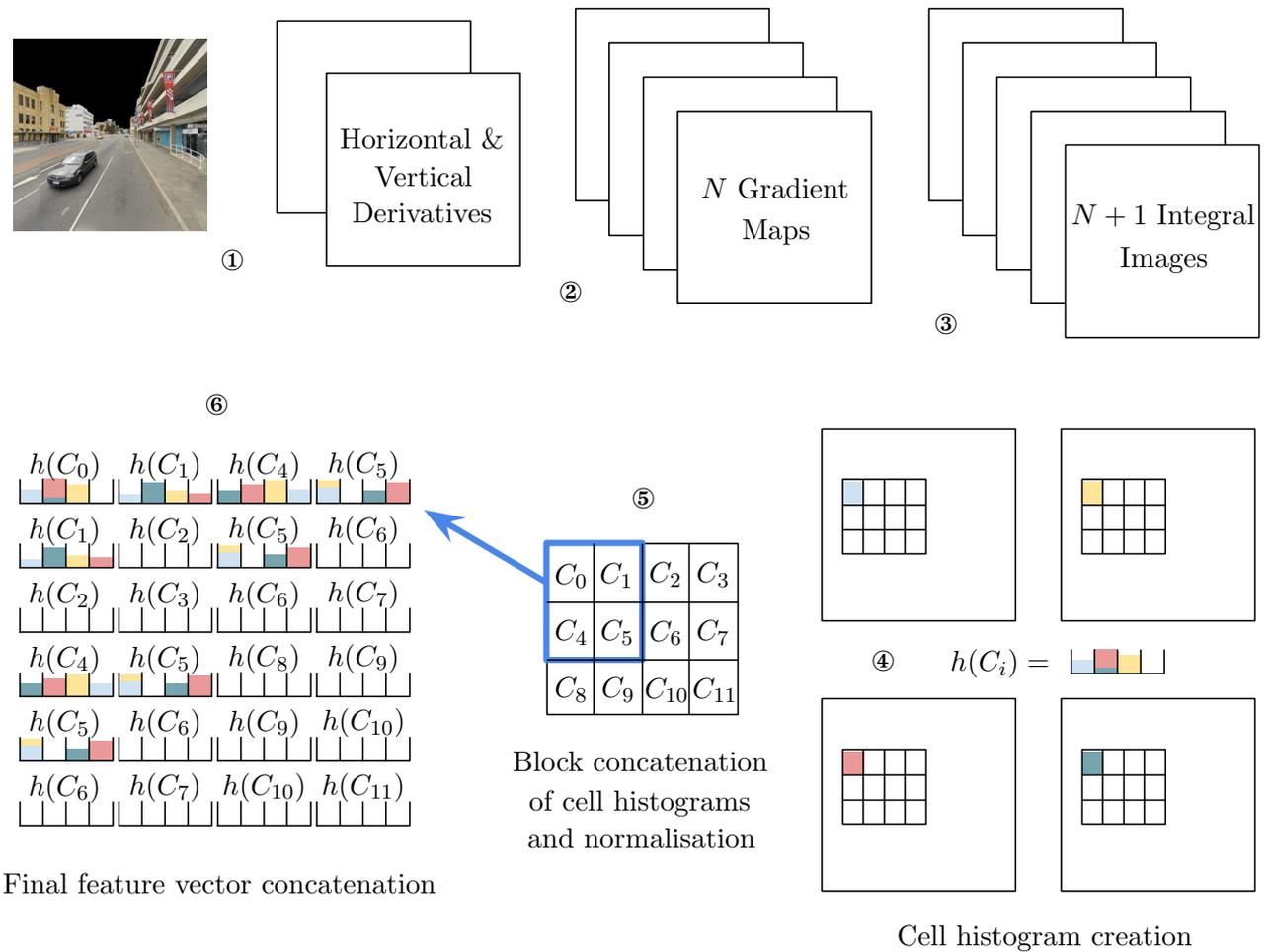


Figure 4-2: The main steps of the Pro-HOG feature extraction algorithm.

processing of the first phase in steps 1 – 3 with the second phase of processing comprised of steps 4 – 6. The steps are outlined as:

1. Generation of the horizontal and vertical derivative maps at pixel resolution.
2. “Binning” of the pixel gradients according to magnitude into N separate image “bins”.
3. Creation of $N + 1$ integral images.

4. Per cell aggregation of pixel gradients from the integral images into length N histograms.
5. Aggregation and normalisation of the cell histograms into 2×2 cell-blocks.
6. Concatenation of the 2×2 cell-block histograms into a final feature vector.

Prior to step four, the pixel dimensions and location of the cell grid within the image are determined. In step 4 of the diagram, the cell grid (over which the features are extracted) is shown as a subregion of the initial image from which the integral images are generated. This is the typical use case for object detection. For classification, the cell grid is coincident with the entire image. The following section explains each of these steps in detail.

4.1.3 Phase One: Pixel Resolution Contrast Gradient Integral Images

As in HOG, square root gamma correction can first be applied to the image during an initial pre-processing stage to attempt to correct the impact of lighting issues that could interfere with accurate contrast measurements. For each pixel value α , the corrected α' is found as:

$$\alpha' \leftarrow \sqrt{\alpha\Gamma} \tag{4.1}$$

where Γ is set as the constant of maximum brightness (nominally 255 for 24 or 32 bit images). Pro-HOG is designed to accept triple byte RGB or intensity (single channel) images, but it can also accept multi-channel images of any intrinsic data type including floating point values. Equation 4.1 is not required for non-appearance based data (such as depth maps).

In HOG, the image extract is initially resized to fixed pixel dimensions given by predetermined parameters that determine one aspect of the eventual size of the feature vectors being extracted. The first phase of Pro-HOG does away with this initial resizing step, and the image extract is processed at the same pixel dimensions as provided. As previously discussed, the motivation for doing this is to avoid the possibility of introducing gradient anomalies as artefacts of image up or down sampling.

Steps One and Two

Every pixel in the input image is initialised with a zeroed out set of gradient maps \mathcal{G} with $|\mathcal{G}| = N$ being the desired number of histogram bins. Horizontal and vertical derivative operators are applied to each pixel in the input image to create separate real valued horizontal

\mathbf{h} and vertical \mathbf{g} difference maps. For clarity, both the gradient maps and the difference maps are represented here as vectors. The contrast orientation of each element (pixel) i is calculated as

$$\theta_i = \arctan \frac{g_i}{h_i} \quad (4.2)$$

and the corresponding contrast magnitude is calculated as

$$\delta_i = \sqrt{g_i^2 + h_i^2} \quad (4.3)$$

For images having more than a single channel (*e.g.* RGB images), the orientation that is used is given by the largest δ_i over the channels. θ_i is mapped to the histogram gradient maps indexed by b_0, b_1 , and b_2 and soft-binned into these in proportion to δ_i . The whole process is given in algorithm 4.1.3 for an image having generated horizontal and vertical difference maps \mathbf{h} and \mathbf{g} .

Algorithm 1 Pro-HOG’s calculation of oriented contrast magnitudes over the vertical and horizontal difference maps of an input image (assumes single channel for simplicity).

```

1: function MAKEGRADIENTMAPS( $\mathbf{g}, \mathbf{h}$ )
2:   for  $g_i \in \mathbf{g}, h_i \in \mathbf{h}$  do
3:      $\theta \leftarrow \arctan \frac{g_i}{h_i}$ 
4:      $\delta \leftarrow \sqrt{g_i^2 + h_i^2}$  ▷ Single channel image assumed
5:      $\phi \leftarrow \frac{N\theta}{2\pi}$ 
6:      $b_1 \leftarrow \lfloor \phi \rfloor \bmod N$  ▷ Centre bin index
7:      $b_2 \leftarrow (b_1 + 1) \bmod N$  ▷ Right bin index
8:      $b_0 \leftarrow (b_1 + N - 1) \bmod N$  ▷ Left bin index
9:      $\mathcal{G}_{b_0}(i) \leftarrow \mathcal{G}_{b_0}(i) + \delta(1 - \phi + b_1)$ 
10:     $\mathcal{G}_{b_1}(i) \leftarrow \mathcal{G}_{b_1}(i) + \delta$ 
11:     $\mathcal{G}_{b_2}(i) \leftarrow \mathcal{G}_{b_2}(i) + \delta(\phi - b_1)$ 
12:   end for
13:   return  $\mathcal{G}$ 
14: end function

```

The divisor in the calculation of θ at line 5 in algorithm 4.1.3 specifies the angular width of each histogram bin in terms of the histogram length N . An angular width of $2\pi/N$ (as used in algorithm 4.1.3) specifies that the contrast orientations should be binned over the full circle – that is, whether contrast is increasing or decreasing (and not just its magnitude) is used to determine the correct bin. Setting this divisor as π/N specifies that contrast orientations should be binned only over half a circle, *i.e.* that the sign of the contrast is irrelevant in determining the histogram bin. For example, $N = 18$ gives 20° per bin if binning over the full 2π radians. Gradient angles may be binned over a π radian range if the direction of gradient change is thought not to be important in the encoding of features. The histogram length N is

one of the factors used to determine the length of the final feature vector. Binning over a full or a half circle does not change the size of the final feature vector, but it changes by a factor of two the angular resolution at which contrast gradients are encoded which can have a large impact on the feature vector's ability to generalise to new data when used in a classification/detection framework. Dalal and Triggs (2005) showed in their original implementation of HOG that binning over a full 360° in the pedestrian detection task decreases detection performance from binning over 180° , although they also state that binning over the full 360° does help with detecting instances of other object classes such as cars and motorbikes (although they do not provide results to demonstrate this).

Pro-HOG does not incorporate cell level bilinear interpolation of the histogram values, which is used in HOG, because the definition of the cell grid is deferred to stage two of the algorithm. In HOG, the cell grid is defined prior to the extraction of pixel contrast gradients (from the resized image) and the histogram values can be bilinearly accrued in proportion to their magnitudes in the histograms of the cells adjacent to the cell currently being processed. In HOG, this is done to further reduce the possibility of cell histograms overfitting to the training data to promote good generalisation (Dalal and Triggs, 2005).

Step Three

The gradient maps resulting from algorithm 4.1.3 are then convolved with a 3×3 Gaussian kernel to further improve the extractor's robustness to slight variations in contrast orientation. Larger smoothing kernels were tried (as well as no smoothing), but this resulted in degraded classification accuracy. In HOG, the size of the Gaussian kernel is given by the fixed cell pixel dimensions. The resulting N smoothed gradient maps are then converted to N integral images. A final integral image is defined as the sum over all of the integral image histograms. This final integral image is used to more efficiently normalise the feature vectors during stage two of the algorithm using only three arithmetic operations (according to the definition of integral image summation) given in figure 4-1 instead of $N - 1$ arithmetic operations.

4.1.4 Phase Two: Cell Histograms and Final Descriptor Generation

The aim in both HOG and Pro-HOG is to associate a histogram of contrast gradients of length N with each cell of a fixed size grid where each cell bounds a group of pixels. In HOG the pixel dimensions of these cells are fixed, and proportionately sized encoding of features is achieved by resizing the input image. In Pro-HOG, the pixel dimensions of the cells are not

explicitly fixed, but the number of cells, and the dimensions of the grid of cells are. Although each cell's pixel dimensions are not fixed, their aspect ratios are congruent with the aspect ratio of the cell grid. The width and height of the grid of cells may be set independently, but for simplicity of explanation here, they are assumed to be equivalent.

Figure 4-3 shows how classic HOG (on the left) first resizes the input image to fixed pixel dimensions before extracting gradient information and defining the cell grid. In Pro-HOG (on the right), the image remains at the original size and contrast gradient information is extracted at the original image size. The pixel dimensions of the cells are determined according to the original image pixel dimensions and the dimensions of the cell grid. The fact that the image is not initially resized means that this step is undertaken more efficiently by Pro-HOG.

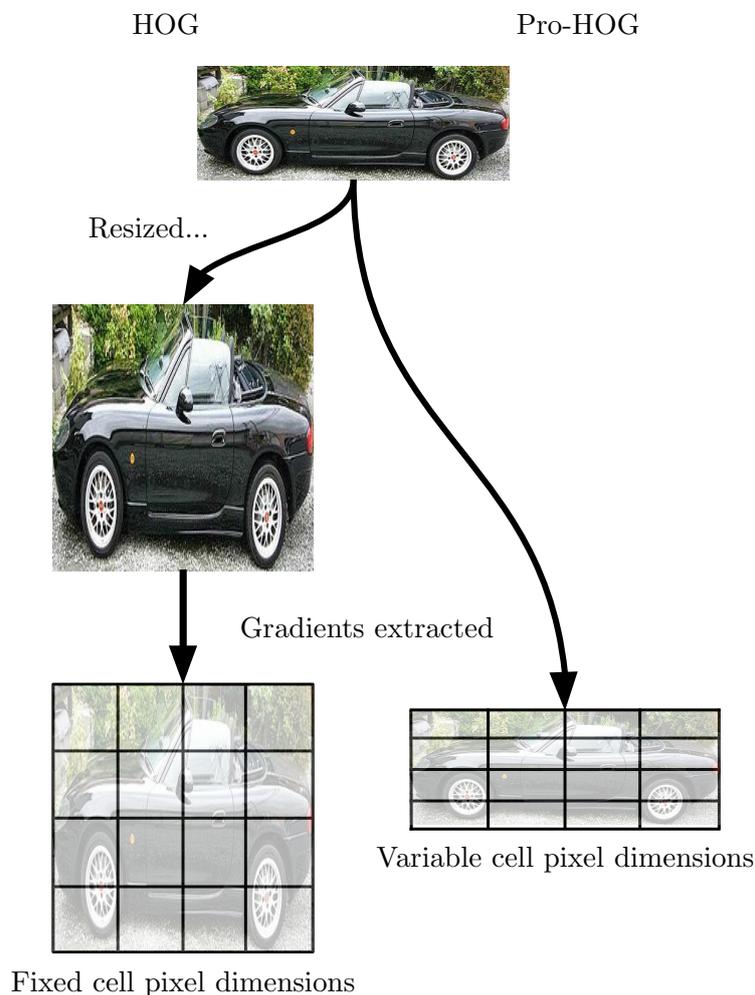


Figure 4-3: Image scaling and cell grid definition for HOG (left column) versus Pro-HOG (right column). Image from the PASCAL VOC 2007 “Car” dataset.

The dimensions μ, ν of the cell grid are predetermined for all extracts, and this introduces a lower limit of 2×2 pixels on the size of the image extracts for which Pro-HOG can construct a feature vector. The pixel dimensions of the image must be no smaller than the predetermined cell dimensions, because the pixel dimensions of each cell must be at least one pixel on a side (though it is not required that cell dimensions be square). The optimum dimensions of the cell grid can be found through cross validation against known data for any particular object classification/detection task. Cell dimensions cannot be too large, or the features will be sampled from the image at too high a resolution which can lead to overfitting of the training data and poor generalisation (increasing the rate of false negative or Type II classification errors). Cell dimensions that are too small however, can generalise too well because not enough discriminative information about the object is recorded in the resulting feature vector, and this can result in many false positives (Type I) misclassifications.

Step Four

The bounding rectangle defining the region of the image to extract features from is supplied to the algorithm (the rectangle may be commensurate with the original image dimensions) and the dimensions of the bounding rectangle are used to define the pixel dimensions of the cells. The pixel dimensions of each cell are set by the proportion of cells that fit vertically and horizontally within the predetermined dimensions of the cell grid overlaid on the image extract (as shown in figure 4-3). When the number of pixels fitting either horizontally or vertically do not divide into whole numbers by the dimensions of the cell grid, the remaining pixels are distributed over a subset of the cells so that cells may not be exactly the same pixel dimensions.

A histogram \mathbf{h} such that $|\mathbf{h}| = N$ is then generated for each of the cells by calculating the sum over the cell region of each of the histogram bin integral images. Given a rectangular cell c with a top left corner x, y having width and height w, h , the general formula for calculating the sum of the values within that area of integral image I is given as

$$sum(c) = I_{x+w-1, y+h-1} - I_{x-1, y+h-1} - I_{x-1, y-1} + I_{x-1, y-1}. \quad (4.4)$$

Figure 4-1 shows this calculation on an example image. It is preferred to define the integral image such that corresponding image values are shifted one pixel down and to the right leaving the first row and column as zeros. This allows for the replacement of the $w - 1$ and $h - 1$ terms with w and h terms in the indexing of the rectangular subregion and equation 4.4 can be written more succinctly. The value of histogram bin h_i is set from the histogram bin

integral image I_i as

$$h_i = I_{i,x+w,y+h} - I_{i,x,y+h} - I_{i,x,y} + I_{i,x,y}. \quad (4.5)$$

The calculation of the integral image subregion always has constant complexity irrespective of the size of the subregion so each cell's histogram can be calculated in an amount of time linear in N .

Step Five

Each cell's histogram is concatenated into 2×2 blocks of cell histograms \mathbf{v} which are then normalised according to the *L1-sqrt* given in Dalal and Triggs (2005) defined as the square root of normalisation by the ℓ_1 -norm plus a small constant to avoid division by zero errors:

$$\mathbf{v} \leftarrow \sqrt{\frac{\mathbf{v}}{\|\mathbf{v}\|_1 + \epsilon}} \quad (4.6)$$

Other normalisation schemes (as used by HOG) such as ℓ_2 -norm normalisation require the sums of the squares of the individual cell histogram values which are more costly to compute since totals can only be generated after the cell histograms have been produced – not during phase one of the algorithm where the ℓ_1 -norm can be more efficiently calculated by summing over all the pixel gradient magnitudes and stored in an integral image. This efficiency is realised primarily during object detection, where many feature vectors are calculated from different (possibly overlapping) subregions of the same scene. In the original evaluation of HOG, Dalal and Triggs (2005) show that using the normalisation scheme given in equation 4.6 does not result in significantly degraded accuracy from using normalisation schemes based on the ℓ_2 -norm. The calculation of the whole feature vector during phase two therefore executes in constant time regardless of cell pixel dimensions.

Step Six

The final feature vector \mathbf{V} consists of the concatenation of the \mathbf{v} feature vectors from all the 2×2 cell blocks in the defined image subregion resulting in a final feature vector length of $|\mathbf{V}| = 4N(\mu - 1)(\nu - 1)$.

The feature vector output from the Pro-HOG extractor for an image can be visualised by splitting each 2×2 cell block's $4N$ feature vector into the individual cell histograms, and using the histogram values at the corresponding image locations to draw lines with grey values that are in proportion with the magnitudes of each of the bins. The lines are drawn

orthogonally to the orientations, and darker lines indicate larger contrast magnitudes in that direction. Figure 4-4 shows a visualisation of the Pro-HOG features extracted from an image of Lena. Figure 4-5 displays a zoomed in section of the image of Lena and the corresponding



Figure 4-4: Lena Pro-HOG visualisation

section of the Pro-HOG visualisation showing how the histogram orientations make a “+” shape over the bridge of the nose and the eyes. The darkness in these regions causing this shape to appear is a result of the strong vertical contrast magnitudes along the nose combined with the strong horizontal contrast magnitudes across the eyebrows. In general, darker regions in the image correspond to larger contrast magnitudes.

4.1.5 Expected Benefits

In the original evaluation of HOG (Dalal and Triggs, 2005), fixed size dimensions for cells of 8×8 pixels gave the best results when being used to classify 1805 64×128 images of persons, using a linear classifier trained with a SVM. It was suggested that the dimensions of the cells were suited to the particular dimensions of the training and validation data such that each cell’s histogram extracted information at a “good” level of detail for the object type. Each cell encoded features for a level of detail commensurate with the ability to discern arms and legs (for instance), but not at so great a level of detail such that irrelevant features of the objects were encoded (*e.g.* folds of clothing, hairstyles, *etc.*) that would act to cause overfitting to the training data. In order to maintain this fixed cell size, images are first resized which

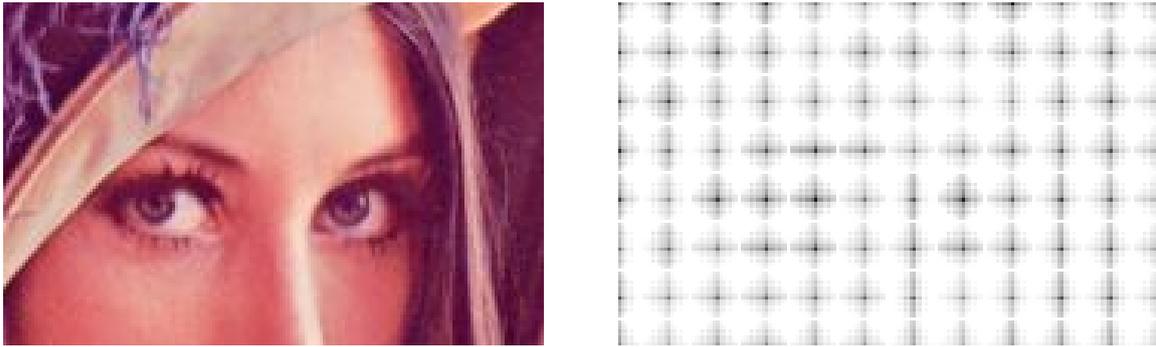


Figure 4-5: A zoomed in image of Lena showing the detail of the eyes in the visualised Pro-HOG image, in particular the contrast magnitude peaks (darker parts) around the eyes and the verticality of the nose bridge given by the strong vertical orientations shown by the histograms in that area.

may possibly introduce artefacts related to up or down sampling the image. In Pro-HOG, no image resizing is conducted and there is no possibility of introducing such artefacts. The dimensions of the cell grid are defined *a priori*, so the benefit of having cell pixel dimensions that are well scaled to a particular object classification/detection task can still be realised, and comparisons between Pro-HOG feature vectors reflects spatial correspondences between objects (given constancy of other variables such as viewpoint and rotation).

In HOG, the use of cells having fixed pixel dimensions means that an image is initially broken down into discrete regions. For classification tasks, this is not problematic because the cell grid will always (after resizing of the image extract) perfectly fit the bounding box of the object. However, this is not the case for object detection where an object may or may not be present in a test image at some unknown position and size. In this case, feature vectors must be extracted from very many different locations in the image and at different sizes. Fixing the cell pixel dimensions then requires construction of scale-space image pyramids if feature vectors are to be extracted at different scales (effectively changing the resolution of the cell grid). A complete set of HOG feature vectors must then be calculated for every region of pixels defining a cell in each image. Calculating the initial pixel level histograms of oriented gradients in Pro-HOG during the first phase of the algorithm is not costly because there is no need to perform normalisation at this point. Normalisation (and calculation of square roots) is only performed as and when cell level HOG features are needed. The normalisation of the histograms in the HOG algorithm can be costly to carry out repeatedly at all levels of an image pyramid. Further, because the positions of the cells are fixed in the test images in the image pyramid, if objects are not located at pixel coordinates that are whole multiples of the cell pixel dimensions, then the objects will be misaligned and spatial correspondence

between the extracted feature vectors from the test image and the feature vectors held in the classifier for the object will be degraded. Pro-HOG feature vectors can be extracted at pixel level resolution (*i.e.* a new Pro-HOG feature vector can be extracted at a position offset from the previous test location by a single pixel rather than a whole cell pixel step). This means that Pro-HOG feature vectors can always be efficiently extracted from any location in a test image, maximising their ability to detect an object at an arbitrary position.

Pro-HOG is implemented as part of a custom generalised feature extraction, learning and evaluation framework. The version of HOG used to test against Pro-HOG is available through the Open Computer Vision (OpenCV) software library; an extensive library of computer vision, image processing and pattern recognition routines that is continually being expanded as new algorithms are introduced by the research community. OpenCV is heavily used in both research and industry, and as a result its algorithms are thoroughly tested and verified to work as expected.

4.2 Testing Pro-HOG’s Parameters

In this section, different aspects of the Pro-HOG algorithm are evaluated and contrasted against the classic HOG algorithm in terms of classification accuracy.

The main dataset used in this section is the PASCAL Visual Object Classes (VOC) 2007 dataset (Everingham *et al.*, 2010). This dataset includes images of people (some examples of which are shown in figure 4-6). Images of people are chosen to classify against because this is the object class primarily used in the original evaluation of HOG by Dalal and Triggs (2005). Dalal and Triggs (2005) introduced a new dataset for testing HOG: the INRIA dataset containing 1805 64×128 pixel images of people cropped from personal photos (examples of which can be seen in figure 4-7). The more recent Pascal VOC 2007 dataset contains 1777 examples of people (not including the examples in the dataset marked as “difficult” or “truncated”) in a much more varied range of poses and scales than in the INRIA dataset, and with the image extracts having varying pixel dimensions instead of being fixed size. This is a better dataset for the purpose of evaluating Pro-HOG because it is based on the idea of encoding features from images at their native pixel resolution, and it allows for the effect of resizing the image extracts to be tested separately to see if image resizing actually does introduce interpolation artefacts that can influence classification accuracy. The Pascal VOC dataset (in its yearly iterations) has been very widely used by researchers across the field of object recognition and scene classification (Lampert *et al.*, 2008; Vedaldi *et al.*, 2009; Divvala *et al.*, 2009; Felzenszwalb *et al.*, 2010; Zhu *et al.*, 2010; van de Sande *et al.*, 2010; Malisiewicz

et al., 2011; Alexe *et al.*, 2010; Girshick *et al.*, 2014). It is of comparable size to the INRIA dataset, and is regarded in general as a very difficult dataset to produce good results over (various versions of the dataset were used in annual object recognition challenges running from 2006 to 2012 (Everingham *et al.*, 2015)).



Figure 4-6: Sample extracts from the “Person” Pascal VOC 2007 dataset. Extracts have varying dimensions and are resized for display here.



Figure 4-7: Example images from the “INRIA” dataset (Dalal and Triggs, 2005)

The Pascal VOC 2007 dataset contains a number of other object classes to test against. In particular, the dataset contains 729 examples of cars (again, ignoring “difficult” and “truncated” examples) which are chosen to test against in addition to the “person” class. Some examples from the “car” dataset are shown in figure 4-8. Cars express much less variability in their morphology compared to people who take on varying shapes according to a range of factors including their age and current activity (which greatly changes the configuration of their parts and orientation with respect to the viewer). As such, the degree of variation in the appearance of cars is more constrained than for people; factors such as colour, texture and viewpoint affect variation in a car’s appearance much more than changes due to age (rusting) or configuration (having the doors, boot or bonnet open or closed). Images of cars more commonly show them upright on the ground (although a few examples in the Pascal VOC

2007 show images of cars from overhead). HOG and Pro-HOG features are extracted over the whole of the image given by the ground truth bounding rectangle for each object extract. This invariably means that feature vectors will encode contextual information about an object too. This is widely accepted as being a good thing for the identification of objects since variation in the appearance of an object’s background is useful for determining the presence of the object itself (Rabinovich *et al.*, 2007; Torralba and Sinha, 2001; Kim and Medioni, 2011). For representations of people, the variation around their strict boundaries is much greater than for cars which are most commonly positioned on roads or other flat texturally homogeneous surfaces. For these reasons, it is expected that for both HOG and Pro-HOG, classification accuracy will be much better for cars than for people even though the person dataset is much larger.



Figure 4-8: Sample extracts from the “Car” Pascal VOC 2007 dataset. Extracts have varying dimensions and are resized for display here.

The pixel dimensions of the images for the “Car” and “Person” datasets vary considerably. Figure 4-9 shows the smallest (on the left) and the largest examples from the dataset. The smallest is 24 pixels wide by 10 pixels tall. The largest is 467 pixels wide by 373 pixels tall.

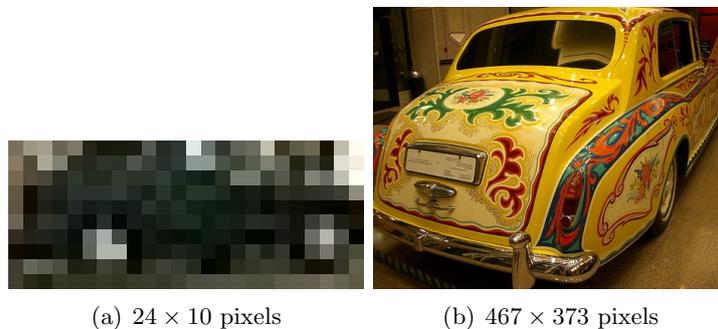


Figure 4-9: Smallest and largest examples from the “Car” Pascal VOC 2007 dataset.

The smallest and largest examples from the “Person” dataset are shown in figure 4-10 with the smallest being 11×29 pixels, and the largest being 439×374 pixels.

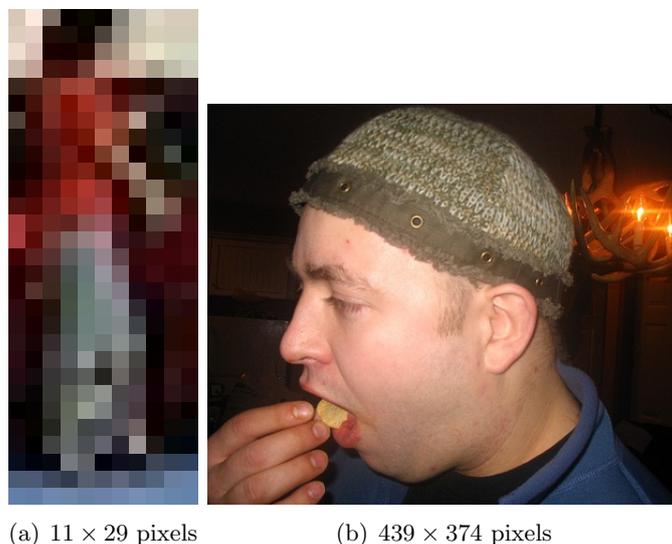


Figure 4-10: Smallest and largest examples from the “Person” Pascal VOC 2007 dataset.

The very large person example shows that some of the examples are not complete instances of the object class in question (even though the dataset provides a labelling of “truncated” examples). Other examples of poor quality instances (truncated or occluded) can be seen in the selection of examples shown in figure 4-6. Surprisingly, the dataset also contains a large number of instances of people riding or standing near bicycles or motorbikes. This might result in classifiers learning structural features about bikes as well as people which may decrease the classification accuracy on this dataset. The “Car” dataset also contains examples of cars that are partially occluded or are atypical in general (including images of cars from unusual viewpoints – in particular, looking down towards the car from above).

4.2.1 General Methodology

Negative examples are collected independently given each object class being tested against. Only images in the Pascal VOC 2007 dataset known not to contain the positive object class are used for the negative extracts. To identify the dimensions of the subregions from these images to use as the negative extracts, the bounding boxes from random positive object instances are used. This is done to avoid the possibility of introducing bias concerning differing aspect ratios between the positive and negative extracts. Initial testing using negative extracts with pixel dimensions having randomly assigned aspect ratios led to unrealistically favourable results being produced for both HOG and Pro-HOG.

As described in section 4.1, Pro-HOG (and HOG in its original design) can also directly parse multi-channel images by selecting the channel from each pixel expressing the largest contrast gradient. In all experiments, the RGB data are first converted to single channel grey scale (intensity) images before processing. This is done to keep the experimental configuration identical for both Pro-HOG and HOG because the OpenCV implementation of HOG only accepts grey scale images.

In the original evaluation of HOG, only linear classifiers trained using SVMs were used (Dalal and Triggs, 2005). In addition to linear classifiers, Pro-HOG and HOG are evaluated in this work using non-linear SVMs. With the radial basis function used in the non-linear SVM (see 2.2.9), the value of γ is set at 0.04. This value was found in testing to give an improvement in accuracy over the linear classifier for nearly all of the feature extractor and dataset pairs tested (to a greater or lesser degree). Values smaller or larger than this were found to be less effective overall.

Other kernel functions that were tried included the polynomial and sigmoid functions (with varying parameters), but neither of these resulted in levels of accuracy comparable to the linear or radial basis functions. The polynomial and sigmoid functions take more than a single parameter and so testing them is a far more involved process. The focus here is not to evaluate the performance of different kernel functions, but only to show that classification in a high dimensional feature space (such as is encoded by HOG and Pro-HOG) may be better served by a non-linear, rather than a linear classifier.

For all experiments, the positive and negative image extracts are randomly divided into five groups for cross validation. Five groups were chosen because this provides a good proportion of data to train the classifier with in each round, while allowing for reasonably efficient collation of results. During testing, using more than five groups did not give significantly

different results to ten-fold cross validation. More extreme cross validation schemes (such as leave-one-out cross validation – training on $N - 1$ data to validate a single instance in each round) were not tried. Randomisation used pseudo random number generators seeded with the same values in each experiment to ensure that each experiment’s results varied only due to the parameters under evaluation.

Each of the five groups is used as the validation set in turn, with the other four groups aggregated together to train the classifier. This five fold cross-validation is conducted for each setting of a parameter in each of the feature extractors. Each round of cross-validation generates a two class confusion matrix for the positive and negative examples. Figure 4-11 shows sample output displaying an example confusion matrix along with other information concerning the configuration of parameters such as the SVM and the current feature extractor being evaluated.

```
SVM Cross Validator
Parsing Pascal VOC directories... done
Loading data for class 'person' with min size (width, height) [25 x 25] finished - count = 1562
Loading data for class 'random' with min size (width, height) [25 x 25] finished - count = 15620
Using data resized to [72 x 72]
Extracting 17182 feature vectors with Pro-HOG [ 4 FALSE 8 8 ] (FV size = 1024) (took 18824.8 msecs)
Validation set sizes (positive, negative) = 312, 3124
Training set sizes (positive, negative) = 1248, 12496
SVM_COST: 1
SVM_EPS: 0.0001
KERNEL: linear
GAMMA: 1
COEFO: 0
DEGREE: 1
Doing 5-fold cross validation . . . . .
[ Results after validation set 5 of 5 ]
```

	Real positives	Real negatives	Totals	
Predicted positives	TP: 1411	FP: 579	1990	PPV (Precision) = 0.7090
Predicted negatives	FN: 4839	TN: 61901	66740	FNR (Type II) = 0.7742
Totals	6250	62480	68730	F1 = 0.3425
	TPR (Recall) = 0.2258	FPR (Type I) = 0.0093		Accuracy = 0.9212

Figure 4-11: Sample output from a single round of cross-validation

The output represents a single data point giving classification accuracy for a given feature extractor and SVM configuration. The displayed statistics include the Recall or True Positive Rate (TPR), the Precision or Positive Predictive Value (PPV), the False Positive Rate (FPR – or type I error rate), the False Negative Rate (FNR – or type II error rate), the F1 statistic which is the harmonic mean of precision and recall, and the overall accuracy (ACC). Details concerning the derivation of these statistics are given in section 3.5.

4.2.2 Gradient Sign Sensitivity

Pro-HOG (and HOG as originally designed) allows for gradient changes to be measured over 360° (sensitive to the sign of the gradient change as well as magnitude), or over 180° (insensitive to the sign of the gradient change – effectively storing the absolute magnitude). For a fixed histogram length, the effect of setting Pro-HOG/HOG to be sensitive to the sign of the contrast gradient has two effects. Firstly, the angular resolution at which gradient magnitudes are stored is halved. That is, each histogram bin is responsible for storing gradient magnitudes over angular sections that are twice as wide as in the case where Pro-HOG / HOG is insensitive to the sign of the gradient change. Secondly, gradient change in a given direction is stored in a bin in the histogram that is $\frac{N}{2}$ bins distant from the bin used to store gradient change in the opposite direction (increase in gradient is treated as different to decrease in gradient).

For some object types, distinguishing between the sign of the gradient change can help to improve classification accuracy by widening (enhancing) inter-class discrimination. However, for other object types, the effect may be to increase intraclass variation within the positive class without necessarily improving inter-class discrimination leading to a relative decrease in accuracy. Even if the length of the histogram is doubled in the case of contrast gradient sign sensitivity (meaning that angular resolutions are equivalent in both cases), the effect of not distinguishing between the gradient change in one direction versus change in the opposite direction will still result in sampling over a different feature space, and thus the class distributions (and the resulting classifiers) will be different. It is instructive and worthwhile to see this difference for a particular object class, by comparing classification accuracy at a given histogram length when Pro-HOG is insensitive to the sign of the contrast gradient, versus the classification accuracy at twice the histogram length when Pro-HOG is sensitive to the sign of the contrast gradient.

The difference in the setting of the contrast Gradient Sign Sensitivity (GSS) parameter is shown in figure 4-12. The diagram shows a single cell of 4×4 pixels with two pixels α and β (marked red and blue respectively). In Pro-HOG, the pixel dimensions of each cell are defined by the cell grid dimensions and the pixel dimensions of the image. In HOG, the pixel dimensions of the cells are defined *a priori* (the nominal setting given by Dalal and Triggs (2005) is 8×8 pixels). The values of the pixels adjacent to pixels α and β are shown. These values are used to calculate $\theta_{\{\alpha,\beta\}}$ and $\delta_{\{\alpha,\beta\}}$ according to equations 4.2 and 4.3. Each cell is associated with a histogram, with length fixed to eight bins in this example. Two histograms are shown underneath the cell. The upper histogram shows how and where in the histogram the two pixels contribute in relative terms when Pro-HOG is set to be insensitive to the sign

of the contrast gradient – *i.e.* dark to light = light to dark. The lower of the two histograms shows how and where these values are aggregated in the histogram when the direction in which contrast is changing is taken into account – *i.e.* dark to light \neq light to dark. Note that the intermediate step of producing the gradient maps according to algorithm 4.1.3 is not shown here. The histograms are generated directly from the gradient maps rather than the original input image.

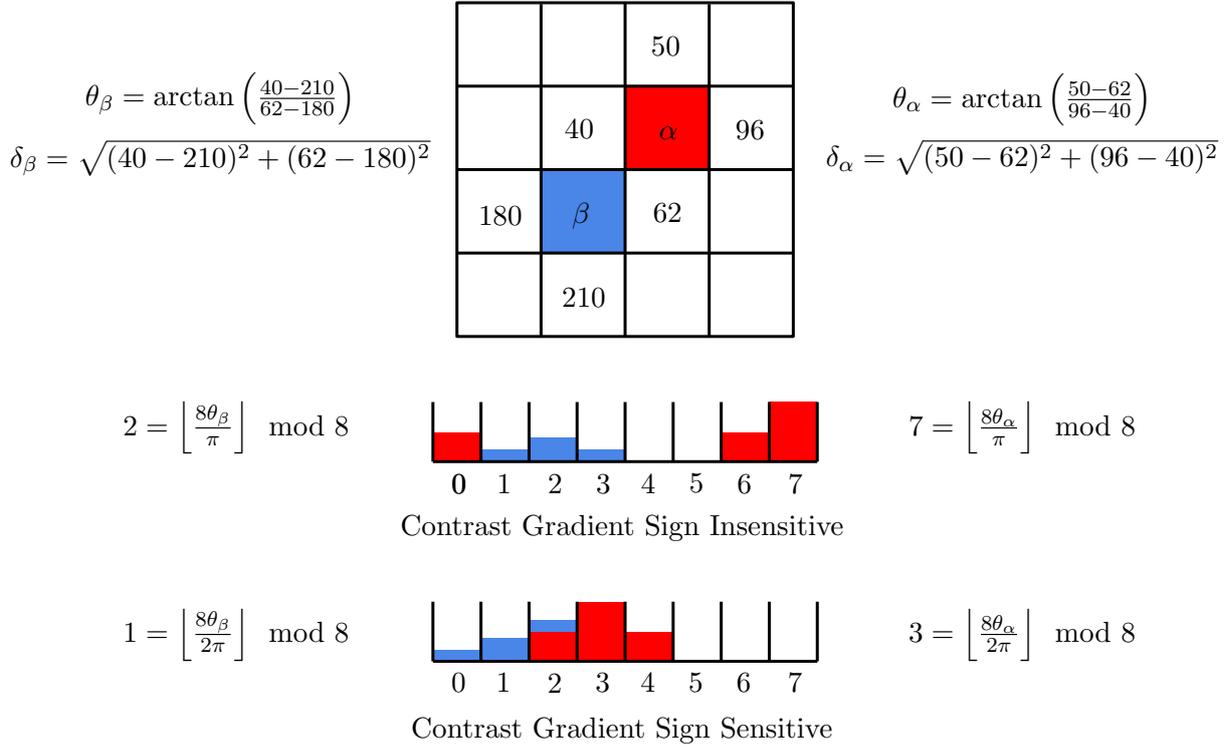


Figure 4-12: Contrast gradient binning inside a single 4×4 pixel cell. The histogram bins into which gradient magnitudes are binned differ depending on whether Pro-HOG is sensitive to the direction of the gradient (the lower histogram) or not (the upper histogram).

Gradient measurements that are diametrically opposite are stored in the same bin when Pro-HOG is insensitive to the sign of the gradient change. This means that even when Pro-HOG uses a histogram that is twice the length when sensitive to the sign of the gradient change, the feature encoding will still be characteristically different (and not just in the length of the generated descriptors which will be twice as long).

The histograms in figure 4-12 show the “soft-binning” of the gradient magnitudes in adjacent bins. In HOG, orientation magnitudes are also aggregated in the histograms of adjacent cells. This is a feature that is not implemented in Pro-HOG due to the separation of the pixel gradient extraction phase and the cell histogram generation phase.

Figure 4-13 shows precision-recall curves giving the classification accuracy of Pro-HOG for the person and car Pascal VOC 2007 datasets with GSS both enabled and disabled. For an otherwise fixed configuration of Pro-HOG, these results show how precision and recall change over a varying range of decision thresholds. Pro-HOG is configured here using a histogram length of nine, and cell-block dimensions of 8×8 (9×9 cells) which emulate the settings of HOG as evaluated in Dalal and Triggs (2005).

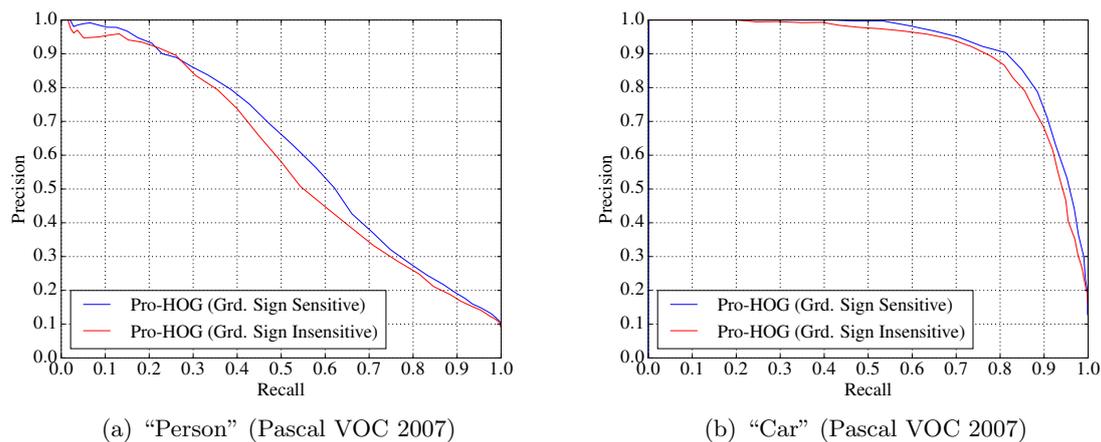


Figure 4-13: Pro-HOG precision-recall curves comparing classification accuracy with gradient sign sensitivity enabled and disabled (Linear SVMs).

For both the person and car datasets, the single configuration precision-recall results show a slight improvement in precision and recall at all decision thresholds when Pro-HOG is configured to be sensitive to the sign of the contrast gradient. In the case of the person dataset, these results are interesting because they conflict with the results of Dalal and Triggs (2005) where in the analysis of HOG, it was found that better accuracy over the INRIA person dataset was obtained when HOG was configured to be insensitive to the sign of the contrast gradient. The rationale given for this was that the wide range in background colours and clothing made the sign of the contrast uninformative. However, the Pascal VOC 2007 dataset arguably represents even more in the way of variation concerning these characteristics (as well as greater variation in other characteristics such as pose and viewpoint).

In the original evaluation of HOG, Dalal and Triggs (2005) state in their methodology that they use left-right reflections of the person data and that the subjects in the INRIA dataset are always upright. Using left-right flipped images in this way acts to remove any possible bias in the more horizontal measurements of contrast orientation. This is appropriate, because people are roughly symmetric in the vertical axis when standing, and the data should not bias one particular orientation (*e.g.* looking to the left or to the right).

To make a fair comparison, a dataset for the “Person” class was created using the 1777 original images in the “Person” dataset plus their left-right reflections resulting in a positive dataset of 3554 images. Again, ten times this number of negative instances were created – also using flipped left-right images to avoid introducing a possible source of bias. The same doubling of data (by including the left-right image counterparts) was performed for the “Car” dataset. The results of performing the classification experiments again using these datasets are shown in figure 4-14.

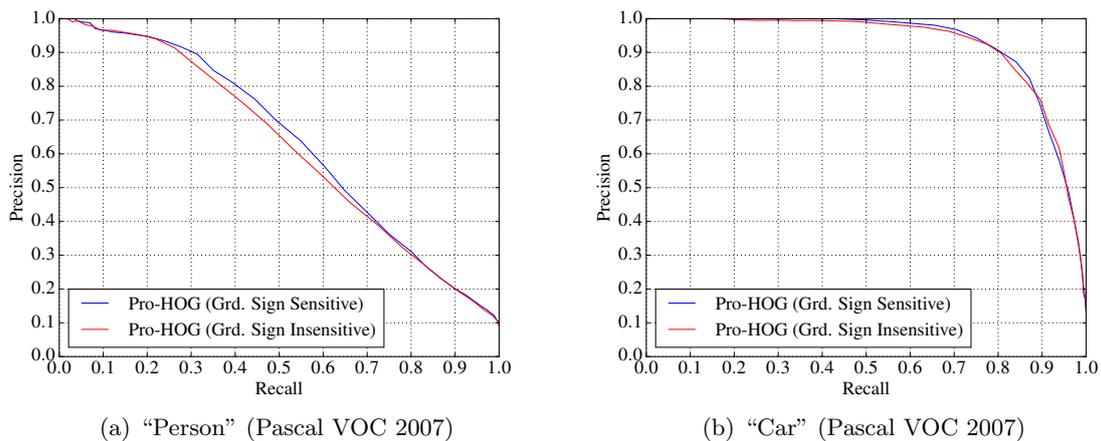


Figure 4-14: Pro-HOG precision-recall curves comparing classification accuracy with gradient sign sensitivity enabled and disabled (Linear SVMs) (*Original images, plus horizontally reflected images*).

Figure 4-14(a) shows that the use of horizontally flipped image counterparts does decrease the disparity in accuracy between the gradient sign sensitive and insensitive versions of Pro-HOG. The GSS enabled version of Pro-HOG still gives slightly better accuracy however – contradicting the explanation given by Dalal and Triggs (2005) for the results of HOG on the INRIA person dataset. In particular, Dalal and Triggs (2005) state that on the INRIA person dataset, HOG performs significantly *worse* when sensitive to the sign of the gradient. The results in figure 4-14(b) show that the discrepancy in accuracy on the “Car” dataset between the two different versions of Pro-HOG is also reduced, but that the GSS enabled version is also still marginally better performing. Further experiments such as repeating the original experiments on the INRIA dataset using Pro-HOG will help to resolve this issue, but are beyond the scope of this thesis.

4.2.3 Histogram Length

The evaluation of the GSS parameter in section 4.2.2 uses a fixed histogram length of nine. This was based on the recommended setting of this parameter as used in HOG (Dalal and Triggs, 2005). This section repeats the experiments of the previous section using a range of histogram lengths. This is done to ascertain if the nominal setting of nine is still appropriate for Pro-HOG when carrying out subsequent experiments over a range of different object types.

The length of each cell’s gradient histogram affects the final length of the feature vector and thus the time taken to extract the feature vector. The accuracy afforded by the feature vectors for classification does not necessarily improve as the size of the feature vectors increases. Generalisation to new data can be degraded if too much spurious information about the training data is encoded in the feature vectors. Two parameters affect the length of the feature vectors in both Pro-HOG and HOG. The first is the parameter that sets the length of each individual cell’s histogram. This affects the angular width of the histogram bins. The wider the angle, the fewer the bins and the greater the number of measurements set in each bin on average. This means that not enough discriminative information about gradient angles can be encoded in the histograms leading to degraded classification precision. If the length of the cell histograms is too great, the feature vectors may start to overfit to the training data since each bin represents a narrower gradient angle. This may degrade classification recall. The second parameter is the number of cell blocks that defines how many histograms are created for an extract. In this section, it is the first of these parameters that is varied to understand the empirical effects on accuracy. The effect of varying the second of these parameters is gauged in section 4.2.5.

In the following experiments, recall and precision accuracy at the default unadjusted classification threshold (trained using both linear and non-linear SVMs) is evaluated. Under evaluation are the “Person” and “Car” Pascal VOC 2007 datasets. For both HOG and Pro-HOG, cell-block dimensions are fixed at 8×8 (cell-block overlap for both HOG and Pro-HOG remains fixed as the width/height of a cell giving cell-grid dimensions of 9×9). For HOG, cell pixel dimensions are fixed at 8×8 pixels. In HOG, images are therefore resized to 72×72 pixels prior to feature extraction. Pro-HOG does not resize the images.

Cross validation is carried out for varying histogram lengths from 2 to 25 bins giving a minimum feature vector length of $4 \times 2 \times 8^2 = 512$, and a maximum feature vector length of $4 \times 25 \times 8^2 = 6400$. The tests are carried out as in section 4.2.2 with both GSS enabled and disabled for Pro-HOG. As previously noted, the reference implementation of HOG being

used in these tests is insensitive to the sign of the gradient. The results of these experiments are shown in figures 4-15 and 4-16.

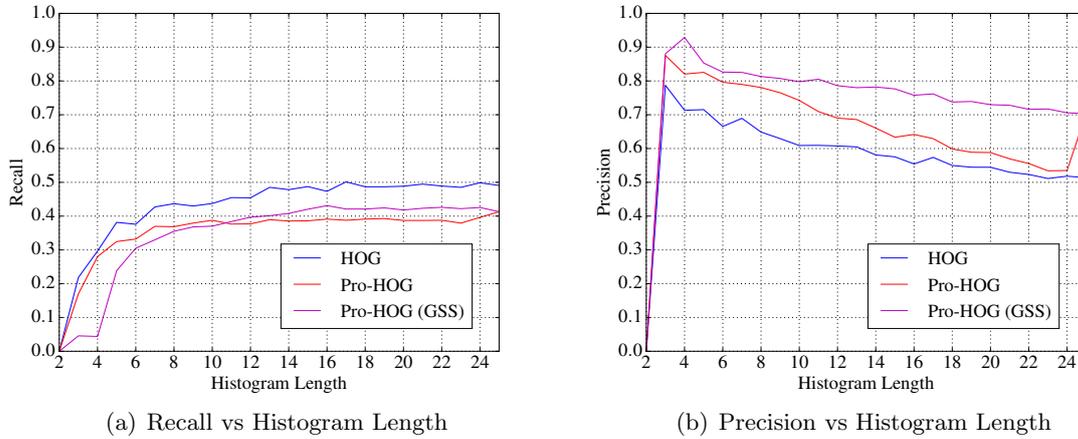
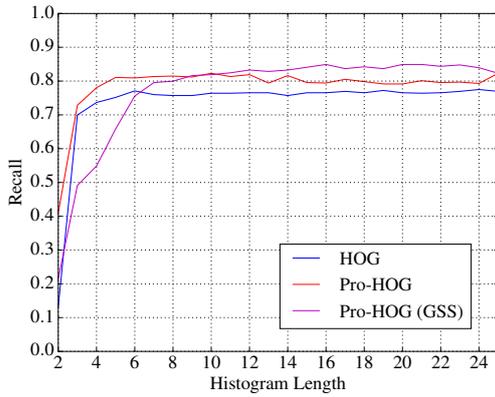


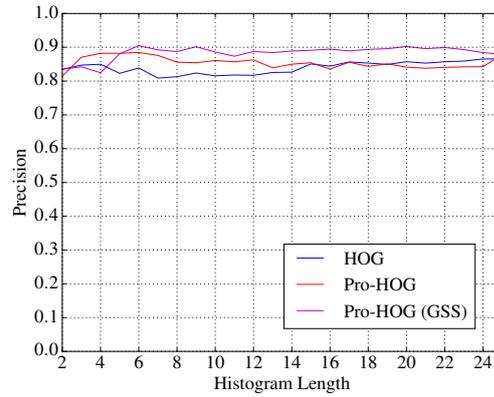
Figure 4-15: Pro-HOG classification precision and recall versus cell histogram length for HOG, and GSS enabled and disabled versions of Pro-HOG. (“Person” Pascal VOC 2007 dataset with linear SVMs).

On the “Person” dataset, figure 4-15 shows that while HOG has better recall accuracy overall (fewer false negatives), Pro-HOG gives better precision (fewer false positives). The improvement in precision by Pro-HOG is greater than the corresponding degradation in recall meaning higher overall accuracy (although higher recall may be preferred over precision in real world use). Pro-HOG with GSS enabled has better recall than with GSS disabled at histogram sizes greater than ten. While recall does not increase for GSS disabled Pro-HOG at histogram lengths larger than ten, recall does increase for GSS enabled Pro-HOG up to a histogram length of 16, albeit slowly. Thereafter, recall no longer improves. For all three classifiers, precision degrades after an initial maximum value at a cell histogram length of three (for HOG and GSS disabled Pro-HOG) and four (for GSS enabled Pro-HOG) although the rate of degradation in precision as the histogram length increases for GSS enabled Pro-HOG is less pronounced than for GSS disabled Pro-HOG. There is an unusually large jump in precision at 25 bins (also reflected to a lesser extent in recall) for GSS disabled Pro-HOG which requires further investigation since this value is so unrepresentative of the trend up until this point.

On the “Car” dataset, figure 4-16 shows that Pro-HOG improves both recall and precision over HOG with linear classification. The higher values for recall and precision for all three of the extraction techniques (compared to the “Person” dataset) are indicative of the ability of the SVM to find a good separating boundary between the positive and negative classes.



(a) Recall vs Histogram Length



(b) Precision vs Histogram Length

Figure 4-16: Pro-HOG classification precision and recall versus cell histogram length for HOG, and GSS enabled and disabled versions of Pro-HOG (“Car” Pascal VOC 2007 dataset with linear SVMs).

As in the case of the “Person” dataset, enabling GSS gives higher accuracy in Pro-HOG at larger histogram lengths (greater than ten for recall, but greater than five for precision), but the improvement in recall and precision is not as great this time. The unusual bump in recall and precision for Pro-HOG is again seen at bin 25, although it is not as pronounced as in the case of the “Person” class. This indicates that the outlier value is probably not a random occurrence and that it is unrelated to a specific collection of data. One possible explanation is that this value is related to the specific pattern of pseudo random numbers generated when performing cross-validation in every experiment (the random number generator is always seeded with the same value).

For all three feature extractor configurations in both the “Person” and “Car” experiments, increasing the cell histogram length and thus the resolution at which orientation information is extracted does not lead to a degradation in recall and a corresponding improvement in precision as expected; the opposite behaviour is in fact observed. In general, classifiers in higher dimensional feature spaces are less accurate if the size of the dataset remains constant because the average density of observations in feature space decreases. However, strong correlations between multiple features may help the classifier to retain high accuracy even as the number of features increases. Since classification accuracy remains high even as the length of the individual cell histograms is increased, it may be that the more important parameter in determining the accuracy of the classifier are the dimensions of the cell grid and the number of individual cell histograms used to form the final feature vectors.

Non-Linear Classification

A linear classifier may not allow for the best separation of feature space given the changing length of the cell histograms and so it is instructive to repeat the experiments using non-linear classification and the radial basis function of equation 2.19. Given the same feature vectors, the non-linear classifier may be better able to separate the positive and negative classes, as demonstrated by improved recall and precision. The results of these experiments using non-linear classification are shown in figure 4-17 for the “Person” dataset, and the results for the “Car” dataset are shown in figure 4-18.

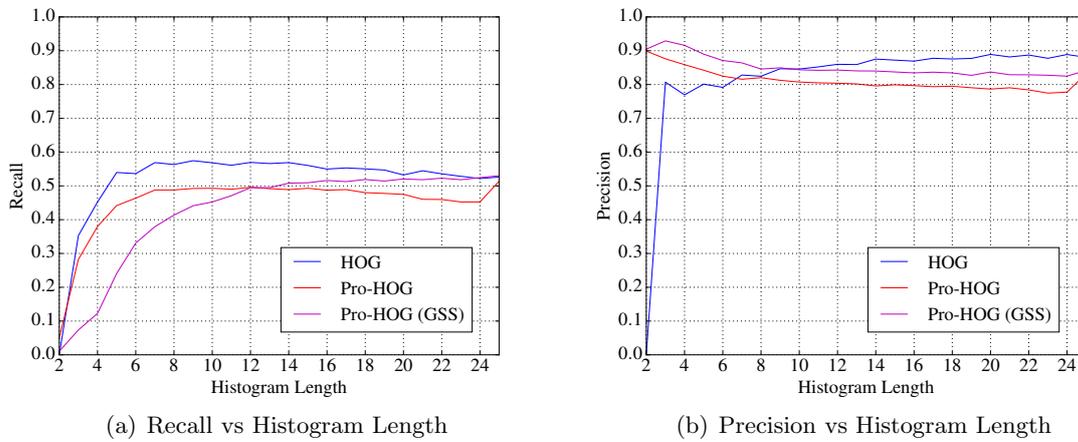


Figure 4-17: Pro-HOG classification precision and recall versus cell histogram length for HOG, and GSS enabled and disabled versions of Pro-HOG. (“Person” Pascal VOC 2007 dataset with non-linear SVMs).

Figure 4-17(a) shows overall that recall is improved over all three feature extractors, but that this improvement is more pronounced at the lower range of histogram lengths. For GSS disabled Pro-HOG and HOG, maximum recall is reached at a histogram length of nine. For GSS enabled Pro-HOG, accuracy in recall monotonically increases over the tested range, although the increase is much more gradual after a histogram length of 16. The recall values for GSS enabled Pro-HOG have magnitudes that approximately match those of GSS disabled Pro-HOG at half the histogram length up to a histogram length of ten (for GSS enabled Pro-HOG). This is congruent with the explanation of how the angular width mapped to each histogram bin in the GSS case only matches the angular width in the GSS disabled case when GSS histograms are twice as long. The increasing improvement in recall for GSS enabled Pro-HOG compared to the gradually decreasing recall accuracy for GSS disabled Pro-HOG after a histogram length of twelve is explained by the ability of GSS enabled Pro-HOG to more distinctly encode contrast orientations that may be important features for classifying the object class under evaluation. The degradation in recall at the longer histogram lengths

for HOG and GSS disabled Pro-HOG may indicate the better separability afforded by non-linear classification showing that a good separation in feature space has been found for a lower feature space dimensionality. With the non-linear classifier, increasing the dimensionality of features (especially since they encode essentially very similar aspects of the object) does not serve to improve recall, and may be causing the classifier to begin to overfit to the training data because of the narrower gradient angles represented by the histogram bins.

The precision results for non-linear classification of the “Person” dataset in figure 4-17(b) are much higher overall, and remain high even at larger histogram lengths (unlike the linear classification case shown in figure 4-15(b)). After bin nine in the non-linear case, better accuracy in precision is delivered by HOG; as recall decreases with increasing histogram length, the proportion of predicted positive instances (the precision) increases. Once again, over the whole range of histogram lengths, GSS enabled Pro-HOG gives better precision than GSS disabled Pro-HOG, but the difference is not as great – even at larger histogram lengths. In the linear classification case, HOG precision at the smallest effective histogram length started relatively low and decreased but in the non-linear case, HOG precision starts near the same value but improves. With Pro-HOG, precision still decreases after initially high values, but the decrease is much more shallow and the rate of degradation decreases as the histogram length increases. This is further evidence that non-linear classification can produce a better fit for the data than linear classification given the same feature space.

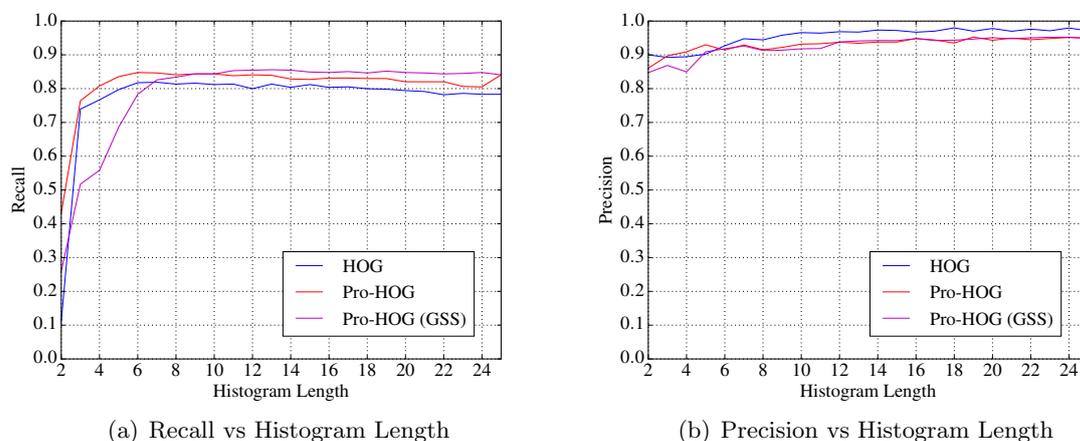


Figure 4-18: Pro-HOG classification precision and recall versus cell histogram length for HOG, and GSS enabled and disabled versions of Pro-HOG. (“Car” Pascal VOC 2007 dataset with non-linear SVMs).

Comparing the results for the non-linear classification of the “Car” dataset (see figure 4-18) to that of the “Person” dataset, similar overall trends in the shapes of the graphs are observed. Recall accuracy plateaus around a histogram length of nine for all three extractors

and thereafter no longer increases and even gradually decreases (this is more clearly seen in the HOG results). Greater dissimilarity between the two cases of classifier is evident in the precision graph of figure 4-18(b). The change from linear to non-linear classification gives a different ordering of the extractors in terms of their relative accuracy. HOG now gives better precision than both the Pro-HOG extractors while there is now very little difference in precision between the two Pro-HOG extractors. Overall precision is higher at all histogram lengths which is congruent with the expectation that non-linear classification gives improved class separability. The average precision taken over all three extractors shows a slight positive correlation with the histogram length (from being negatively correlated in the linear classification case).

Summary

From the above results on both the “Person” and “Car” Pascal VOC 2007 datasets, the following conclusions can be made about the effect of varying the cell histogram lengths in HOG and Pro-HOG. Increasing the length of the histograms does not significantly degrade classification recall, but it is also not improved. Recall accuracy is on the whole higher using non-linear classification. GSS enabled Pro-HOG tends to show better recall than GSS disabled Pro-HOG, but only at larger histogram lengths.

Non-linear classification allows for much improved precision over linear classification. In the linear case, precision is not improved by increasing the length of the histograms and in the non-linear case, precision is only improved in some cases and not significantly as histogram length increases.

For any of the extractor configurations, a cell histogram length no shorter than nine is recommended to maintain sufficiently high classification recall and precision, but no larger than this to avoid the possibility of degradation in accuracy (primarily in precision). This is congruent with the cell histogram length recommendation for HOG in Dalal and Triggs (2005). However, it is not clear if these conclusions are generally applicable or if they are specific to the two datasets tested here. Further testing on a greater variety of object types should help to clarify this.

Given the lack of very significant degradation in classification accuracy as the length of the cell histograms increases, it is probable that this aspect of the algorithm is less important in determining how accurately objects are encoded (for the purposes of classification), than the cell resolution parameter which determines the number of histograms used to produce the final descriptor. The effect of varying this parameter is investigated in section 4.2.5.

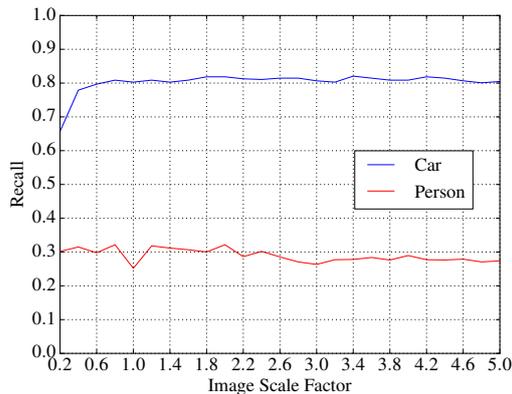
4.2.4 Extract Scaling

In HOG, extracts are always resized to fixed pixel dimensions prior to calculating the cell histograms. This means that all images, unless they are already at the predefined pixel dimensions, experience some degree of resizing. Since the act of resizing an image can modify the quality of the information it portrays (see section 2.1.2), it is important to understand how the act of image resizing prior to the extraction of features affects the quality of the extracted features. A proxy measure for this quality is classification accuracy.

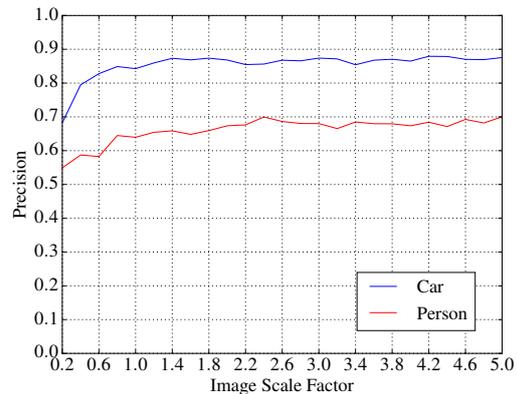
In these experiments, Pro-HOG is evaluated using a fixed configuration against the Pascal VOC 2007 “Person” and “Car” datasets. The individual extracts are resized to be larger or smaller than the originals by different scaling factors. Pro-HOG always generates the cell histograms using the image at the dimensions it is supplied in and so repeating the same classification experiment with differently scaled image extracts will show if the act of scaling the images changes the nature of the extracted feature descriptors, and thus the classification accuracy.

Because the images from within the datasets are of varying sizes, any differences in classification accuracy will be due solely to the effects of scaling the extracts. If the image scale factor is not important, the results ought to show relatively unchanged recall and precision accuracy for the two datasets and no obvious correlation with image scale. Any variation from a consistent response will indicate that scaling the images (either up with factors < 1 or down with factors ≥ 1) acts to improve or worsen classification accuracy. If accuracy does vary with different scaling factors, the results should indicate if there is a preferred scaling factor for the images. It is hypothesised that image scaling (whether up or down) can corrupt the image quality, and so the expectation is that the best classification accuracy will be achieved when no resizing is conducted (a scale factor of 1).

Pro-HOG is configured to use a fixed histogram length of nine and cell-block dimensions of 8×8 (giving cell grid dimensions of 9×9). Each image extract is scaled at intervals of 0.2 by factors from 0.2 to 5.0, and five-fold cross validation is performed using the same methodology as in section 4.2.2. The minimum size of a valid example for both datasets is set to 45×45 pixels so that at the smallest scale factor of 0.2, the images are still large enough to be encoded by the 9×9 cell grid (each cell then represents a single pixel). Extracts smaller than these dimensions from either of the datasets are not used. This results in total dataset sizes of 507 “Car” examples and 1219 “Person” examples. For both datasets, ten times as many negative examples are used. Since the type of classification is irrelevant here, only linear classifiers are used. Figure 4-19 shows precision and recall results for the two datasets.



(a) Recall vs Extract Scale



(b) Precision vs Extract Scale

Figure 4-19: Recall & Precision against Extract Scale using Pro-HOG and Linear Classifiers

These results show that for the “Car” dataset, image scaling by a factor ≥ 1 does not change the quality of the extracted features such that classification accuracy is degraded – either in terms of recall or precision. This is because above a scale factor of one, the graphs remain flat. At scaling factors < 1 , recall and precision quickly degrade.

For the “Person” dataset, scaling the extracts by factors > 1 does appear to show a very slight negative correlation with recall, but a very slight positive correlation with precision. At scaling factors < 1 , only precision is degraded – recall remains relatively unchanged. As the image scale factor increases (> 1), the difference in recall and precision is due to a change in the relative proportion of type I and type II errors (false positives and false negatives). The overall number of true positive predictions remains fairly constant. The anomaly in recall at a scale factor of one is difficult to explain since the same instances were used in each round of cross validation. This was ensured by the use of the same seed for the pseudo random number generator being used. It is probable that this result was simply due to the specific nature of the features extracted at this scale. Similar recall accuracy is seen at a scale factor of three. It is possible that there is no real correlation between accuracy and scale factor – simply that the variance in the data is larger than in the “Car” dataset meaning that there is greater variation in the results.

Summary

Down sampling the extracts at the smallest scaling factors tends to reduce both recall and precision (although this is not clearly seen here in the results on the “Person” dataset). This is not unexpected since down sampling the extracts from their original sizes reduces the amount

of information present. These findings are congruent with those of Dollar *et al.* (2014).

These results help to validate the design of Pro-HOG in its avoidance of scaling input images prior to feature extraction. Image scaling appears to have no overall positive impact on classification accuracy, and if images are scaled down from their provided dimensions, the resulting classification accuracy can be severely degraded. The fact that image scaling increases computation time for no apparent benefit means that there is no good reason to incorporate it into a feature extractor if it can be avoided. HOG resizes all images to fixed pixel dimensions before extracting features. This risks reducing classification accuracy if those images are originally at larger dimensions than the fixed pixel resizing dimensions used by HOG. These results strongly suggest that the extraction of HOG type features without first resizing image extracts should be preferred.

4.2.5 Cell Grid Dimensions

This set of experiments repeats those of section 4.2.2 but use a fixed cell histogram length for all three feature extraction configurations while varying the dimensions of the cell grid over which the histograms are generated. The aim here is to understand how classification accuracy behaves with the changing dimensions of the cell grid, and whether there is any significant difference between Pro-HOG and HOG in this respect.

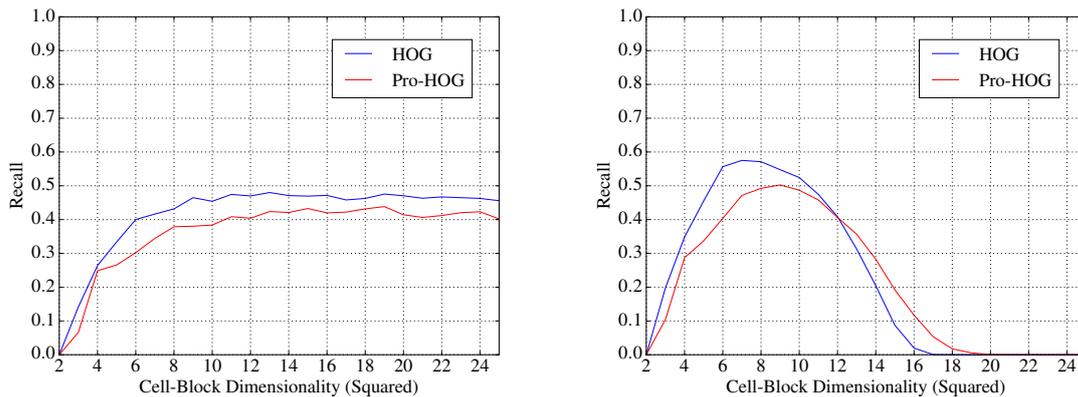
Changing the dimensions of the cell grid affects the relative area of an image encoded by each cell's histogram. Larger dimensions of the cell grid means a denser sampling of the image features with more detail concerning the change across the image / object as a whole. This can result in the descriptors encoding information that is too specific to the individual instances and may lead to poor generalisation past a certain point. Smaller cell grid dimensions means that less detail is extracted and generalisation should be improved, but this may also cause an increase in the false positive rate if the cell grid dimensions are too small. The cell grid dimensions increase in width and height by one cell in each step and so the length of the produced feature vectors increases quadratically (as opposed to linearly when testing the histogram length), so the point where recall begins to degrade may be reached sooner than when solely increasing the length of the gradient histogram.

Neither HOG or Pro-HOG can accept images with pixel dimensions that are lower than the predefined dimensions of the cell grid. In HOG, this is not an issue because images are resized to fixed pixel dimensions prior to processing; images that are too small are up-scaled to the necessary pixel dimensions. Since each data point needs to be generated using the same size

dataset, in Pro-HOG images having pixel dimensions smaller than the cell grid dimensions being tested are first up-scaled so that they can be processed at those cell-grid dimensions. This is acceptable because the evaluation in section 4.2.4 shows that up-scaling images does not degrade the ability of the feature extractor to encode discriminative information about the instances. This also allows the full dataset to be used for each object type. Note that while Pro-HOG only ever up-scales images, HOG may down-scale some input images to the predefined fixed pixel dimensions, and this can cause information to be lost – leading to diminished classification accuracy.

Cell grid dimensions are evaluated from 2×2 cells-blocks to 25×25 cell-blocks. Since each cell-block is 2×2 cells, the cell grid dimensions are one more than this in both height and width (3×3 cells to 26×26 cells). The length of the individual cell histograms is fixed at nine since the results of section 4.2.2 show that this gives good recall and precision over the “Car” and “Person” Pascal VOC 2007 datasets. Larger histograms may result in improved recall and precision but the gain is much smaller and relatively shorter feature vectors also have the practical benefit of being faster to compute. This is also the histogram length recommended for use in the original evaluation of HOG by Dalal and Triggs (2005) (and it is the default parameter setting in the OpenCV reference implementation of HOG). Pro-HOG has gradient sign sensitivity (GSS) disabled to allow for a fair comparison with HOG.

The results showing the effects on recall of varying the cell grid dimensions on the “Person” Pascal VOC 2007 dataset are shown in figure 4-20. Figure 4-21 shows the effects on precision. The experiments are conducted using both linear, and non-linear classification.



(a) Recall vs Cell Grid Dimensions (Linear SVM) (b) Recall vs Cell Grid Dimensions (RBF SVM)

Figure 4-20: Recall versus cell grid dimensions for the “Person” Pascal VOC 2007 dataset

Figure 4-20 shows that Pro-HOG and HOG encode information about the extracts differently as the cell grid dimensions increase. Figure 4-20(a) shows that while HOG has higher recall

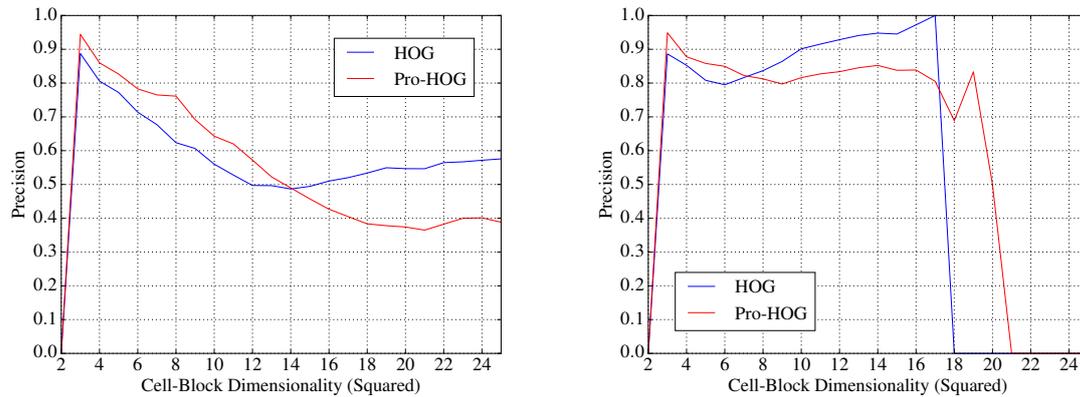
over the full range of cell dimensions tested, the difference in recall accuracy with Pro-HOG is not consistent. The rate of recall is initially similar between the two extractors before diverging at cell-block dimensions of 5×5 . Recall reaches a peak at cell-block dimensions of 11×11 for HOG and cell-block dimensions of 15×15 for Pro-HOG. This difference in behaviour in recall accuracy is more pronounced for the non-linear classifier as shown in figure 4-20(b). In this non-linear case, HOG has a slightly narrower peak in maximum recall around cell-block dimensions of 7×7 . The peak in recall for Pro-HOG comes later at cell-block dimensions of 9×9 .

The larger variation in recall accuracy shown by HOG (especially in the non-linear case) shows that HOG is more susceptible than Pro-HOG to changes in the setting of the cell-grid dimensions. This can be explained by the fact that HOG implements more involved methods of “smoothing” the values of the generated cell histograms. At smaller cell grid dimensions this will act to improve generalisation. However, at larger cell grid dimensions, HOG’s interpolating of adjacent cell histograms may increase the “noisiness” of the generated descriptors, decreasing recall accuracy faster than Pro-HOG which is better able to retain structural details because it does not interpolate histogram values from adjacent cells.

The drop in recall accuracy for both Pro-HOG and HOG at the higher cell-block dimensions seen in figure 4-20(b) is almost certainly due to the classifier overfitting to the training data at higher feature vector dimensionality. The dimensionality of the feature space increases quadratically with the dimensions of the cell grid while the number of observations remains constant leading to a much more sparsely populated space for training the classifiers. In the non-linear case, the greater degrees of freedom available to fit the separating hyperplane means that the learning algorithm is much more likely to support a boundary that is less representative of the true population boundary than in the linear case which is more limited in its placement of the boundary in the same space.

The graph of precision using linear classification in figure 4-21(a) shows a distinct difference in how Pro-HOG and HOG behave under the same configuration. At lower cell-block dimensions, Pro-HOG exhibits higher precision than HOG until cell-block dimensions of 14×14 . Thereafter HOG gives improved precision. A similar trend is seen in figure 4-21(b), but the cell-block dimensions at which HOG provides for improved accuracy comes earlier at 7×7 .

Measurements of precision become much less reliable at higher cell-block dimensions in the non-linear classification case as the overall number of positive predictions falls (both true positives and false positives) *i.e.* the differences in precision are much more pronounced in relative terms. In the non-linear case of precision in figure 4-21(b), the drop off in precision



(a) Precision vs Cell Grid Dimensions (Linear SVM) (b) Precision vs Cell Grid Dimensions (RBF SVM)

Figure 4-21: Precision versus cell grid dimensions for the “Person” Pascal VOC 2007 dataset

comes at the point where the classifiers are failing to make any positive predictions – the differences between the positive and the negative examples are completely opaque to the non-linear classifiers at this high feature dimensionality given the sparsity of the training data.

The results for the “Person” dataset show that HOG is more susceptible to changes in cell-block dimensionality (with slightly narrower peaks, and more pronounced inflections in the non-linear recall results), while Pro-HOG is slightly more stable but has less overall recall accuracy. The higher rates of precision and the lower rates of recall seen with Pro-HOG, especially at lower cell-block dimensions, could be because a larger proportion of the images are not being scaled down as they are in HOG, and also that no interpolation or cell resolution smoothing of the histograms is undertaken by Pro-HOG. Pro-HOG therefore has access to more information in an image on average than HOG at the lower cell-block dimensions and the individual cell histograms will be more specific to the examples they are generated from. This increased detail seen by Pro-HOG in the larger examples will inhibit the recall of those examples. As a proportion, if Pro-HOG is better able to correctly classify the smaller examples but most of the larger examples are classified as negatives, this will show up as increased precision over the positive predictions. The relative stability in recall shown by Pro-HOG over HOG is then evidence of the bias of Pro-HOG towards these smaller examples.

Figures 4-22 and 4-23 show the results of these experiments repeated on the “Car” Pascal VOC 2007 dataset.

As with the recall results for the “Person” dataset, the recall results for the “Car” dataset shown in figure 4-22 show less variance over a greater range of cell grid dimensions for

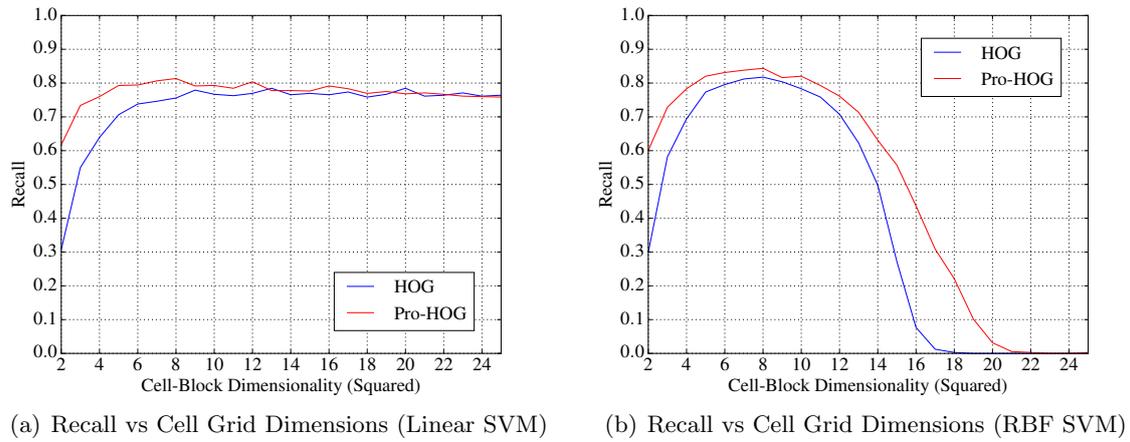


Figure 4-22: Recall versus cell grid dimensions for the “Car” Pascal VOC 2007 dataset

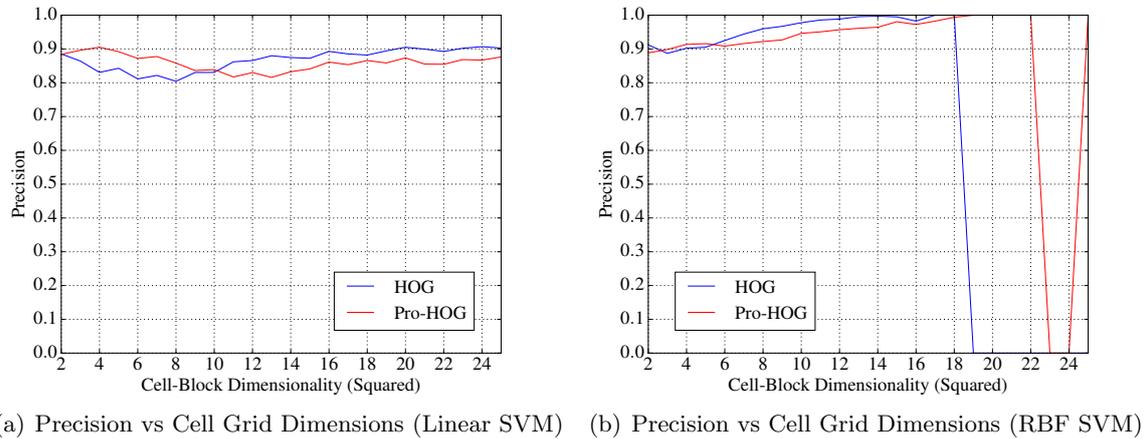


Figure 4-23: Precision versus cell grid dimensions for the “Car” Pascal VOC 2007 dataset

Pro-HOG, but more variance for HOG. Unlike the results on the “Person” dataset, recall is on average higher for Pro-HOG. The difference in precision as the cell grid dimensions vary as shown in the graphs of figure 4-23 is not as great as for the “Person” dataset. HOG has slightly higher precision on average than Pro-HOG and this indicates increased certainty in the positive predictions – a lower relative number of false positives. But with the lower recall this suggests that HOG is overfitting to the training data more than Pro-HOG on this dataset. This is unexpected given that HOG implements methods to interpolate and smooth the characteristics of its generated histograms more than Pro-HOG does. The effect of such functionality is usually to assist in the ability of the feature descriptors to generalise more readily to new observations – increasing recall accuracy at the cost of precision. The fact that this is not occurring on this particular dataset indicates that the lack of these features in Pro-HOG may not place it at too great a disadvantage.

Summary

Given these results on the “Person” and “Car” Pascal VOC 2007 datasets, there is no preferred setting of the cell-block dimensionality that works best in both cases – either for Pro-HOG or for HOG. Both HOG and Pro-HOG have similar classification accuracy on average although they clearly behave differently on the two datasets. On the “Person” dataset, HOG gives improved recall though not necessarily better precision. On the “Car” dataset, Pro-HOG gives better recall, but better precision only at lower cell-block dimensions.

In general, to avoid the possibility of overfitting to the training data (particularly in the case of non-linear classification), lower cell-block dimensions should be preferred. Cross-validation on sample data can be performed to discover feature extraction parameters that are best suited to the particular object classification task.

The results so far show on the two object classes evaluated (the “Person” and “Car” Pascal VOC 2007 datasets), that it is very difficult to empirically establish whether Pro-HOG offers improved recall and precision accuracy in general over HOG. Testing the extractors using only two different object types is insufficient to provide a reliable insight into the behaviour of Pro-HOG as it compares to HOG. In the following section, a more involved set of experiments on a range of different object types from two new datasets is undertaken.

4.3 Evaluation Against Many Object Types

In this section, Pro-HOG and HOG are evaluated for their object classification accuracy over several object types from the AAM and Earthmine datasets (which are described in chapter 3). The feature extractors are used to generate descriptors from both the RGB images (which are first converted to intensity images), and the depth maps that are available in these two datasets.

4.3.1 Experimental Aims

The primary aim of this section is to compare recall and precision accuracy over a relatively diverse range of object types so that some reliable conclusions can be made about the general performance of Pro-HOG compared to HOG in the context of object classification.

The secondary aim is to test if the features that can be extracted by Pro-HOG and HOG from the depth maps available in the AAM and Earthmine datasets are comparable to the features generated from the colour images in these datasets for the purposes of object classification.

Depth Descriptors

To produce feature vectors from the depth data, the extraction algorithms must be able to handle the different data types. The depth data for the AAM and Earthmine datasets are stored as single channel floating point values. Only depth values up to one hundred metres are used. Larger values are presumed to be too distant to prove useful for classification purposes and they are ignored. The depth data must be converted to a format that the extractors are able to parse. The OpenCV implementation of HOG can only process byte length integer pixel values (from 0 to 255 inclusive). In order to use the depth data in the OpenCV implementation of HOG, the images are converted to this format, reducing the precision of the depth values to $\frac{1}{2} \times \frac{100}{255} \approx 0.2$ metres. For the AAM data in particular where depth values are captured at millimetre precision, this is a significant degradation in precision. Pro-HOG is designed to convert all input data to floating point values before constructing the pixel resolution contrast histograms. Although this means that Pro-HOG is slower than HOG in this step (since floating point arithmetic is inherently slower than integer arithmetic), it does mean that the precision of the data values remains intact. Pro-HOG thus allows for

a more accurate encoding of the depth data compared to HOG and should give improved classification accuracy – especially with the AAM object types.

Gradient Sign Sensitivity

Alongside HOG, Pro-HOG is tested against each object type in four different configurations. In two of these, Pro-HOG is configured to be sensitive to the sign of the contrast gradient (GSS enabled). The results of section 4.2.2 show that Pro-HOG gives better classification accuracy on the Pascal VOC “Person” and “Car” object types with GSS enabled. The results on the Pascal VOC 2007 “Person” dataset contradict the rationale given by Dalal and Triggs (2005) for the explanation of why HOG gives better accuracy on the INRIA person dataset when insensitive to the sign of the contrast gradient. Testing the effect of GSS being enabled or disabled over a broader range of object types will help to understand how much of an effect this setting has more generally. It is expected that enabling GSS in Pro-HOG will allow for improved classification accuracy over most, if not all of the object types. As already noted, the OpenCV implementation of HOG used in this work does not allow for HOG to be configured to be sensitive to the sign of the contrast gradient, and so comparisons concerning the effectiveness of GSS are made only between the two different configurations of Pro-HOG for each object type.

Image Resizing Effects

Pro-HOG’s default mode of operation is to extract features from images as they are presented at their native resolution. However, feature extraction using Pro-HOG can be modified to emulate HOG’s behaviour of resizing input images to fixed pixel dimensions prior to the extraction of features. This is undertaken for both the GSS enabled and disabled versions of Pro-HOG. Fixing the dimensions of the input images for Pro-HOG to match HOG’s resizing dimensions means that any differences in classification accuracy will be due to algorithmic differences such as the block-normalisation scheme and the lack of bi-linear cell interpolation of histogram values in Pro-HOG. In particular, this means that the GSS disabled version of Pro-HOG should very closely match HOG but for these few algorithmic differences.

Pro-HOG seeks to trade its lack of some aspects of the HOG algorithm (and resulting possible degradation in classification accuracy), for scale-invariant feature extraction which avoids image resizing, especially image resizing images down from their original dimensions which section 4.2.4 showed decreased classification accuracy. By controlling this aspect of the algo-

rithm and testing over a range of different object types with differing image dimensions, it can be determined whether this trade off is to the overall benefit or detriment of Pro-HOG.

For object types represented at generally smaller sizes than the fixed resizing dimensions used by HOG, it is expected that Pro-HOG will not perform as well as HOG because images will be scaled up which was shown in section 4.2.4 to not decrease classification accuracy. HOG will be able to leverage the aspects of its algorithm that are lacking from Pro-HOG in these cases. For object types with native image sizes generally larger than the resizing dimensions used by HOG, it is expected that Pro-HOG will allow for improved classification accuracy because HOG will downsize the images causing information to be lost. Pro-HOG's scale independent feature extraction should allow for better feature representation and therefore improved classification accuracy in these cases.

4.3.2 Methodological Parameters

Dalal and Triggs (2005) use cell-block dimensions of 8×8 in their evaluation on the INRIA dataset. In section 4.2.5 these dimensions also resulted in reasonable levels of recall and precision accuracy for the Pascal VOC 2007 "Person" and "Car" object types – for both HOG and Pro-HOG. In the experiments in this section, HOG uses fixed cell pixel dimensions of 8×8 – the same as used by Dalal and Triggs (2005) in HOG's original evaluation against the INRIA person dataset. This setting of the cell pixel dimensions may not be ideal for all object types, but it provides a baseline in order to compare against Pro-HOG. This setting results in fixed image resizing dimensions of 72×72 pixels prior to feature extraction since cell-blocks are 2×2 cells and the cell grid resolution must be 9×9 to give 8×8 cell-blocks.

Given the results of section 4.2.2 all datasets are evaluated using the left-right horizontal reflections of the images (both the positive and negative image sets) to avoid any lateral viewpoint bias in the object representations. The size of the negative image set for each class is set at ten times the size of the positive image set. For each object type, the experiments are repeated using both linear and non-linear (RBF) SVMs. For ease of comparison, results for the Pascal VOC "Person" and "Car" classes are also given although only results for the descriptors derived from the RGB images are available because depth information is not available in the Pascal VOC dataset.

4.3.3 Object Types

Table 4.1 shows the breakdown of the object types evaluated giving the number of examples per class (left-right mirror pairs), the minimum and maximum dimensions (in pixels), and the median area (in square pixels) of the examples in each class.

Dataset	Object	Count	Min Size	Max Size	Median Area
Pascal VOC	Car	729	24×10	467×373	14884 sq. pxls
Pascal VOC	Person	1777	11×29	439×374	10816 sq. pxls
AAM	Car	729	11×9	162×78	1089 sq. pxls
AAM	Traffic Light	1172	8×24	235×243	2116 sq. pxls
AAM	Triangular Road Sign	163	9×14	73×204	529 sq. pxls
AAM	Truck / Van	188	19×16	216×113	2025 sq. pxls
AAM	Road Light	1461	14×49	248×347	7921 sq. pxls
AAM	Telegraph Pole	937	5×16	79×272	2304 sq. pxls
AAM	Rectangular Road Sign	668	8×12	99×213	900 sq. pxls
AAM	Generic Road Sign	1489	6×20	95×191	729 sq. pxls
Earthmine	Car	52	16×15	329×169	8100 sq. pxls
Earthmine	Garbage Bin	122	7×9	51×90	289 sq. pxls
Earthmine	Traffic Light	178	8×22	108×318	1156 sq. pxls
Earthmine	Parking Sign	143	7×31	36×169	1225 sq. pxls
Earthmine	Traffic Cone	52	5×9	31×52	400 sq. pxls

Table 4.1: Object Classes used for Object Classification Evaluation

The experiments on each object type are separated into their own sections. For each object type, a representative selection of examples are shown together with the distribution of the extract sizes shown as a histogram. Any pertinent information concerning the object type and how classification on the type may perform given its characteristics is also discussed.

The results are given in the form of tables showing the recall, precision and F1 scores for HOG and each configuration of Pro-HOG, for both the linear and the non-linear classifiers at the default classification threshold. Precision-recall graphs for the two classifiers types used (linear on the left, non-linear on the right) show the resulting accuracy for the different feature extractor configurations over all classification thresholds. Results are shown separately for the descriptors generated from intensity images, and for those generated from depth maps.

Pseudo random number generators with the same seeds determine the subsets of the data in each of the cross validation rounds, so that the results for the RGB and the depth descriptors are directly comparable.

Following the results for each object type, a brief individual summary is given, concentrating

on the behaviour of the different configurations of the feature extractors, and the differences in classification accuracy of the intensity and depth image based descriptor types.

After all of the individual evaluations of the different object types, section 4.4 summarises the findings to make an assessment of the general behaviour of HOG versus Pro-HOG in its different configurations over all fifteen of the tested object types. Some follow up experiments are conducted to clarify questions raised in the individual object evaluations.

4.3.4 Pascal VOC “Car”

Some example images from the Pascal VOC “Car” dataset are given in figure 4-8. Figure 4-24 shows that the size distribution of these examples is large, but only 266 of the 729 examples (36%) are less than the fixed resize dimensions of 72^2 square pixels. This means that almost two thirds of the data are resized to be smaller than their original dimensions – potentially degrading the quality of feature encoding in these cases. It is expected that Pro-HOG in its originally conceived scale independent mode (not emulating HOG’s resizing behaviour) should allow for an overall improved level of classification accuracy over image resizing configurations of Pro-HOG as well as the original HOG algorithm.

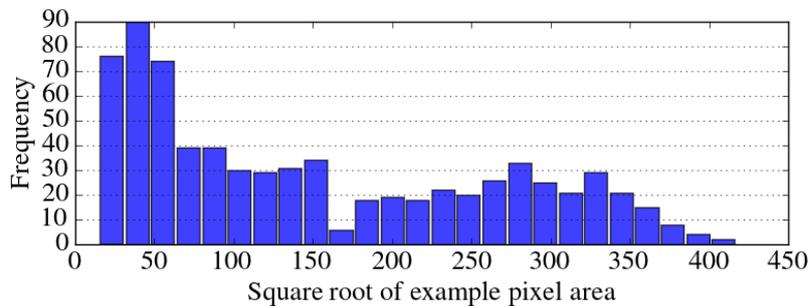


Figure 4-24: Distribution of example sizes for the Pascal VOC “Car” object type.

In the linear classification case (figure 4-25(a)), the hypothesised behaviour is borne out with some caveats. The GSS disabled, image resizing version of Pro-HOG, is most similar in behaviour to HOG, and gives better performance than HOG even though it lacks some of the algorithmic normalisation aspects of HOG. This level of accuracy cannot be entirely explained in terms of the image resizing. The GSS *enabled*, image resizing version of Pro-HOG gives accuracy that is more akin to (and not significantly different from) the two best performing versions of Pro-HOG – both of which do not resize the images.

For the two versions of Pro-HOG that parse the images at their original size, whether or not GSS is enabled, there is very little difference in classification accuracy. However, when

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8411	0.7811	0.8137
Pro-HOG (GSS, 72×72)	Linear	0.8949	0.8121	0.8515
Pro-HOG (72×72)	Linear	0.8511	0.7922	0.8206
Pro-HOG (GSS)	Linear	0.8972	0.8203	0.8570
Pro-HOG	Linear	0.8780	0.8196	0.8478
HOG	Non-Linear	0.9631	0.8409	0.8979
Pro-HOG (GSS, 72×72)	Non-Linear	0.9307	0.8381	0.8820
Pro-HOG (72×72)	Non-Linear	0.9267	0.8237	0.8722
Pro-HOG (GSS)	Non-Linear	0.9243	0.8464	0.8836
Pro-HOG	Non-Linear	0.9297	0.8525	0.8894

Table 4.2: Classification Results – Pascal VOC “Car” (Intensity descriptors)

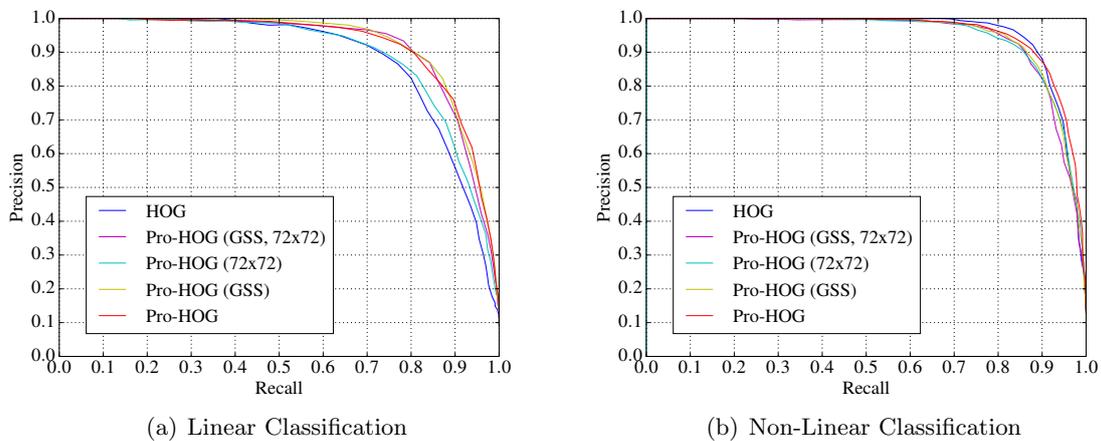


Figure 4-25: Precision versus Recall using feature descriptors extracted from intensity images of the Pascal VOC 2007 “Car” object type.

the images *are* resized to fixed dimensions, enabling GSS improves accuracy – primarily in precision which improves approximately twice as much as recall at the default classification threshold. In the non-resizing versions of Pro-HOG, GSS also mainly improves precision, but the gain is not as large.

In the results for the non-linear classification case shown in figure 4-25(b), the results across the different extractors are much tighter. HOG gives better overall accuracy, closely followed by the GSS disabled version of Pro-HOG (which gives the best overall recall).

Even though enabling GSS does not increase the dimensionality of the feature space, it does allow for greater variance in the range of histogram encodings. With a fixed size of training data but a more flexible training algorithm (as allowed by the non-linear kernel function), this

may be contributing to a less accurate estimation of the classification boundary by the SVM. At the default classification threshold, as seen in table 4.2, resizing the images in Pro-HOG causes a slight drop in recall with no significant change in precision (a drop of $\approx 3\%$ points in recall for the GSS disabled version). In terms of overall accuracy, in the non-linear case, Pro-HOG appears to offer no significant improvement over HOG for this object type.

4.3.5 Pascal VOC “Person”

Figure 4-6 shows some examples from the Pascal VOC “Person” dataset. Like the Pascal VOC “Car” dataset, the size distribution of these data is also large, but with a greater mass of the data at the smaller image sizes. Most of the images are represented at sizes larger than the fixed resize dimensions. In this case, 66% of the images are larger and will be downsized by HOG and the image resizing versions of Pro-HOG – leading to possible information loss. It is expected that Pro-HOG in its original configuration will allow for improved accuracy because of this bias towards larger native image sizes.

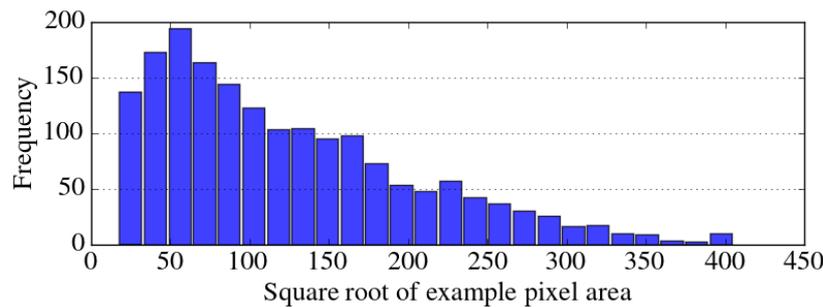


Figure 4-26: Distribution of example sizes for the Pascal VOC “Person” object type.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.6904	0.4316	0.5312
Pro-HOG (GSS, 72×72)	Linear	0.7988	0.4133	0.5448
Pro-HOG (72×72)	Linear	0.7818	0.4114	0.5391
Pro-HOG (GSS)	Linear	0.8247	0.3745	0.5151
Pro-HOG	Linear	0.8001	0.3751	0.5107
HOG	Non-Linear	0.8446	0.6165	0.7128
Pro-HOG (GSS, 72×72)	Non-Linear	0.8244	0.5656	0.6709
Pro-HOG (72×72)	Non-Linear	0.8104	0.5869	0.6808
Pro-HOG (GSS)	Non-Linear	0.8715	0.4961	0.6322
Pro-HOG	Non-Linear	0.8462	0.5478	0.6651

Table 4.3: Classification Results – Pascal VOC “Person” (Intensity descriptors)

Figure 4-27(a) shows that Pro-HOG enables better accuracy than HOG in all four of its

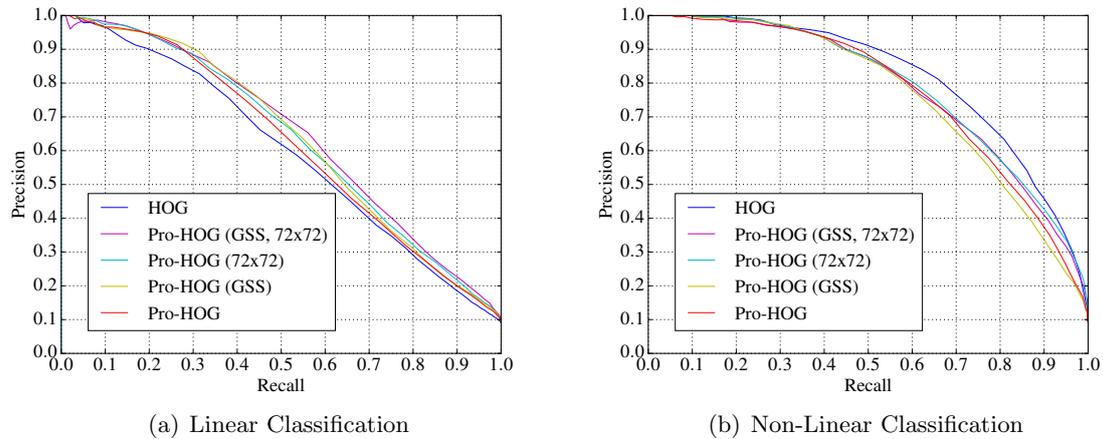


Figure 4-27: Precision versus Recall using feature descriptors extracted from intensity images of the Pascal VOC 2007 “Person” object type.

configurations in the case of linear classification. The image resizing versions of Pro-HOG perform better than the non-resizing variants. Most of this improvement in accuracy is in recall which offsets a slight reduction in precision.

This behaviour is not reflected in the Pascal VOC “Car” class in the linear classification case. There are a much larger number of examples in this class however, and the intraclass appearance variation is also much greater. Down-sizing the images may help to decrease the amount of intraclass variation inherent in the different sizes of images caused by the inherently fractal nature of the detail present in examples of this object type. That is, images at larger resolutions will continue to introduce more detailed structure not present at lower resolutions (*e.g.* clothing textures) instead of simply clarifying the structural information already present (as in the case of objects such as cars that express relatively larger areas of uniform texture and contrast). These results indicate that for certain kinds of objects, depending on whether higher recall or precision is preferred, downsizing the images so that information is lost may *improve* classification accuracy.

In the non-linear case, figure 4-27(b) shows that classification accuracy is again improved overall for all five of the extractor techniques. Pro-HOG in all four of its configurations is worse than HOG in this case however. The better performing versions of Pro-HOG are the image resizing versions (as seen in the linear) case, but the ability of the non-linear classifier to produce a more accurate classification boundary is enhanced by the added benefit of HOG’s cell histogram spatial normalisation and cell grid resolution Gaussian smoothing.

4.3.6 AAM “Car”

Some examples from this class are shown in figure 4-28. As is the case for most of the images in this dataset, the act of projecting the coloured LAS point clouds and extracting the resulting images, does not allow for high quality object representations. Additionally, the viewpoint of the imaging camera (in the middle of the road) and the fact that only stationary vehicles are present in the data, means that cars are located some distance from the camera (in car parks, or parked on the side of the road). These issues mean that the vast majority of the objects are relatively small and indistinct. The size distribution of the data is shown in figure 4-29 which shows that most of the data (98%) are less than the fixed resizing dimensions required for HOG. Therefore, Pro-HOG’s ability to parse the images at their native dimensions should not contribute to improved classification accuracy and Pro-HOG is expected to do no better in this case than HOG (and may be worse given Pro-HOG’s less involved histogram normalisation scheme).



Figure 4-28: Sample extracts from the AAM “Car” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

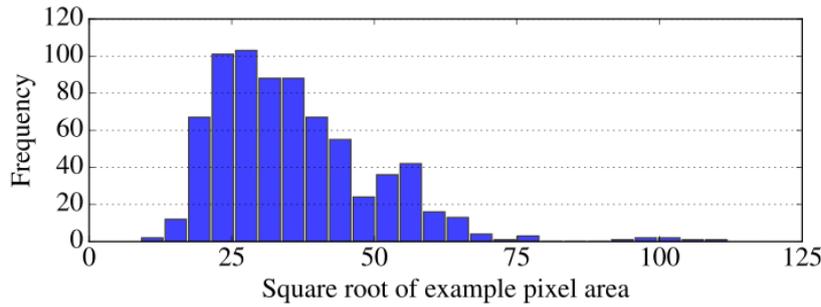


Figure 4-29: Distribution of example sizes for the AAM “Car” object type.

Intensity Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8907	0.8436	0.8665
Pro-HOG (GSS, 72×72)	Linear	0.9293	0.8745	0.9011
Pro-HOG (72×72)	Linear	0.9012	0.8567	0.8783
Pro-HOG (GSS)	Linear	0.9373	0.8717	0.9033
Pro-HOG	Linear	0.9033	0.8717	0.8873
HOG	Non-Linear	0.9632	0.8621	0.9099
Pro-HOG (GSS, 72×72)	Non-Linear	0.9676	0.8813	0.9225
Pro-HOG (72×72)	Non-Linear	0.9510	0.8649	0.9059
Pro-HOG (GSS)	Non-Linear	0.9567	0.8779	0.9156
Pro-HOG	Non-Linear	0.9565	0.8738	0.9133

Table 4.4: Classification Results – AAM “Car” (Intensity descriptors)

Contrary to expectations, in both the linear and the non-linear classification cases, figure 4-30 shows that Pro-HOG (in all four of its configurations) does not allow for classification accuracy that is significantly better than HOG (but not definitely worse either). In particular, the hypothesised behaviour is directly contradicted in the linear case (see figure 4-30(a)), where the two best performing versions of Pro-HOG are the GSS enabled versions – with the non-resizing version having slightly better overall accuracy due to improved precision. Accuracy is very slightly reduced when the extracts are resized using Pro-HOG. In the version of Pro-HOG most similar to HOG (the image resizing, GSS disabled configuration), accuracy in both precision and recall is slightly better than HOG, implying that in this case, classification benefits from *not* implementing bi-linear cell histogram interpolation and cell grid resolution smoothing. This could be because Gaussian smoothing with a relatively larger kernel (as is performed using HOG) further corrupts already noisy data which might better be served by the use of a median filter to remove excess noise.

In the non-linear classification results shown in figure 4-30(b), accuracy is higher overall than

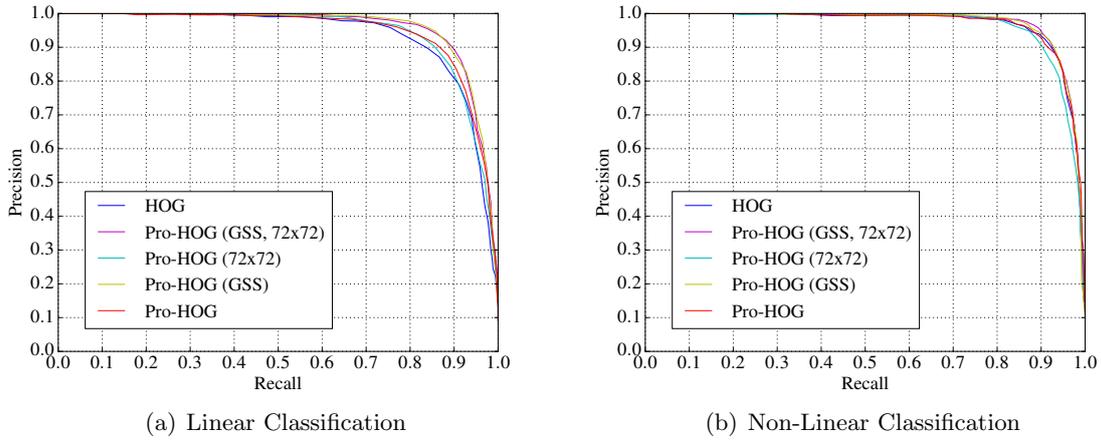


Figure 4-30: Precision versus Recall using descriptors extracted from intensity images of the AAM “Car” object type.

in the linear classification case. The overall ordering of the Pro-HOG extractors is unchanged in terms of the F1 score at the default classification threshold shown in table 4.4, even though the individual precision and recall rankings are slightly different. HOG gives better accuracy than the Pro-HOG extractors in the non-linear case, but the image resizing configurations of Pro-HOG give comparable accuracy. HOG’s recall is still lowest overall, but its precision improves by more than seven percentage points indicating far fewer false positives than in the linear case. This indicates that instead of simply generalising better, HOG allows the SVM to create a classifier that better discriminates between the positive and negative distributions as indicated by the higher precision scores for all of the extractors – not just HOG.

Depth Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9491	0.9211	0.9349
Pro-HOG (GSS, 72×72)	Linear	0.9509	0.9170	0.9337
Pro-HOG (72×72)	Linear	0.9453	0.9122	0.9284
Pro-HOG (GSS)	Linear	0.9513	0.9108	0.9306
Pro-HOG	Linear	0.9315	0.9053	0.9183
HOG	Non-Linear	0.9827	0.9355	0.9574
Pro-HOG (GSS, 72×72)	Non-Linear	0.9804	0.9287	0.9539
Pro-HOG (72×72)	Non-Linear	0.9810	0.9204	0.9498
Pro-HOG (GSS)	Non-Linear	0.9760	0.9218	0.9481
Pro-HOG	Non-Linear	0.9796	0.9232	0.9506

Table 4.5: Classification Results – AAM “Car” (Depth descriptors)

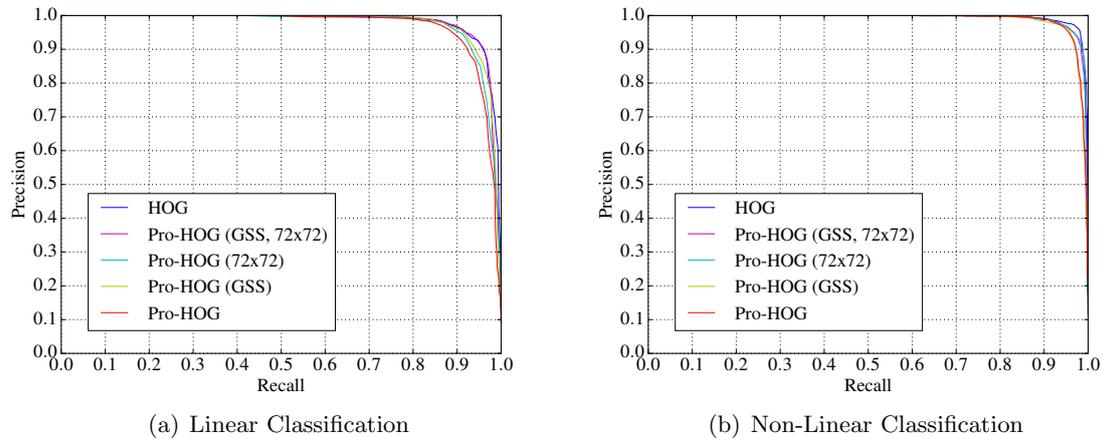


Figure 4-31: Precision versus Recall using descriptors extracted from depth images of the AAM “Car” object type.

With the depth based descriptors, classification accuracy as shown in figure 4-31 is improved in both the linear and the non-linear cases over the intensity based descriptors. This can partly be explained by the poor quality of the colour mapping in the AAM data. That the depth descriptors give such an improvement over the intensity descriptors is surprising given the apparently less descriptive nature of the data. The benefit of the depth data in the AAM dataset is that it is far less susceptible to representational issues that can impair the quality of the RGB data (such as lighting or intraclass variance caused by texture or colour differences that are irrelevant to the object type). This qualitative difference in the data modalities may mean that depth data is inherently more suitable in some contexts for generalised object classification.

The results also show that although HOG is not able to encode depth values as accurately as Pro-HOG (due to the rounding of values that occurs as part of the conversion of the floating point data to integer valued data), HOG gives slightly better accuracy compared to the closest performing Pro-HOG variant (with GSS enabled and resizing the extracts). This could be because there is spurious variation in the surface morphology for this object type which is being encoded by Pro-HOG but not by HOG because its rounding of these values smooths out the depth variations that are not indicative of the object’s type. Pro-HOG in all of its configurations gives very similar results, although the image resizing variants offer the best accuracy in terms of recall in the linear classification case. For non-linear classification, there are few differences in accuracy between the different variants.

4.3.7 AAM “Traffic Light”

Some examples from the AAM “Traffic Light” dataset are shown in figure 4-32. This dataset contains traffic lights consisting of the posts they are attached to as well as the actual cowling and lights. The cowling and lights themselves are generally consistent in morphology, but the lights are not necessarily always attached to simple upright posts (as seen in the second example) meaning that the position of the lights (and the distinctive dark cowling) is not always located in the upper middle of the image. The thinness of the object type means that in some cases the object is difficult to distinguish from the background (see the third example). Figure 4-33 shows that the bulk of the data ($\approx 83\%$) are at a smaller size than the fixed resizing resolution.

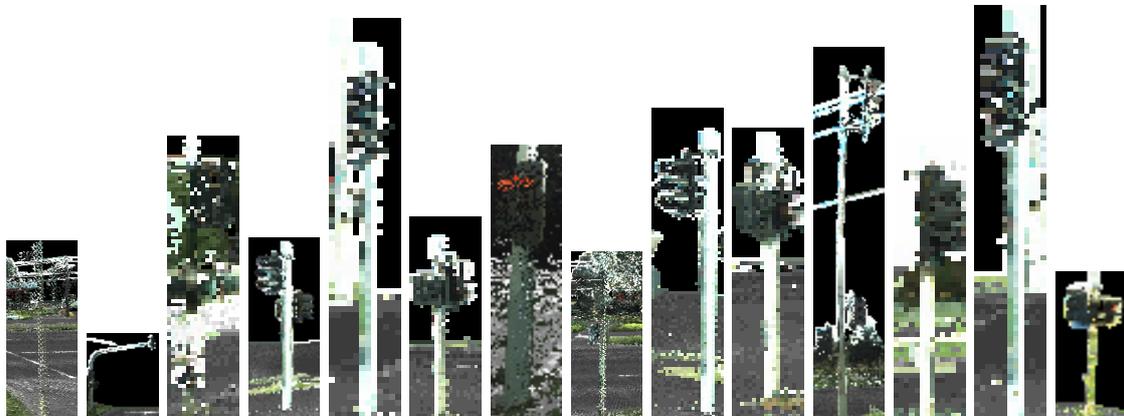


Figure 4-32: Sample extracts from the AAM “Traffic Light” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

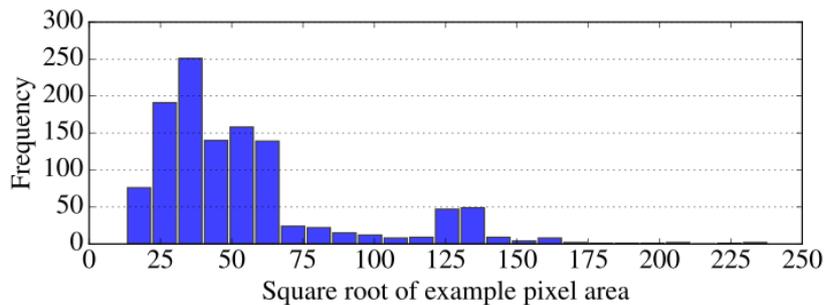


Figure 4-33: Distribution of example sizes for the AAM “Traffic Light” object type.

Intensity Based Descriptors

Figure 4-34 shows that in both the linear and the non-linear classification cases, better accuracy is achieved by the non-resizing versions of Pro-HOG (both the GSS enabled and disabled

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8131	0.6015	0.6915
Pro-HOG (GSS, 72×72)	Linear	0.8840	0.6049	0.7183
Pro-HOG (72×72)	Linear	0.8568	0.5947	0.7021
Pro-HOG (GSS)	Linear	0.8838	0.6391	0.7418
Pro-HOG	Linear	0.8601	0.6476	0.7389
HOG	Non-Linear	0.9470	0.6476	0.7692
Pro-HOG (GSS, 72×72)	Non-Linear	0.9431	0.6156	0.7450
Pro-HOG (72×72)	Non-Linear	0.9522	0.6463	0.7700
Pro-HOG (GSS)	Non-Linear	0.9436	0.6497	0.7696
Pro-HOG	Non-Linear	0.9393	0.6536	0.7708

Table 4.6: Classification Results – AAM “Traffic Light” (Intensity descriptors)

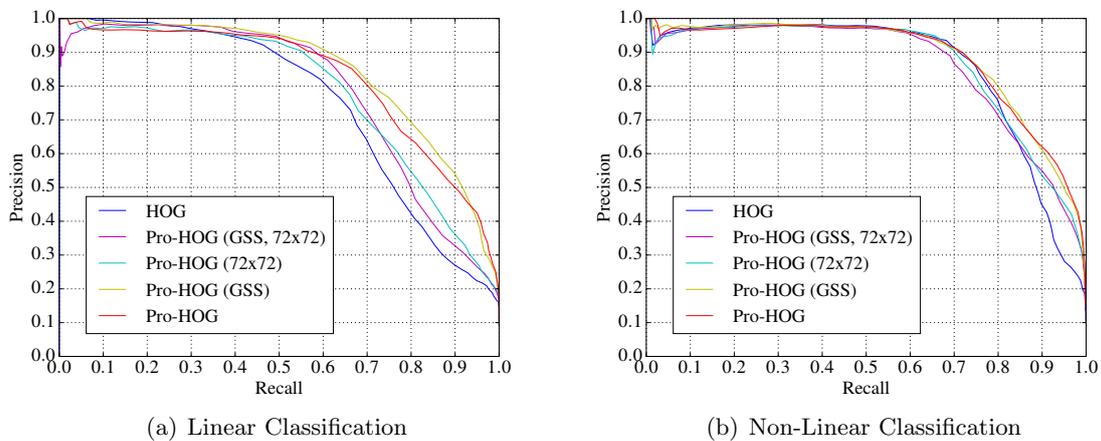


Figure 4-34: Precision versus Recall using descriptors extracted from intensity images of the AAM “Traffic Light” object type.

configurations). The GSS disabled versions improve recall over precision, but the GSS enabled versions trade improved precision for recall. Resizing the images to fixed dimensions reduces precision and recall accuracy. Since the bulk of the images are smaller in size than the fixed resizing dimensions, this provides possible evidence that scaling the images up is contributing to a degradation in accuracy. However, in section 4.2.4, it was demonstrated that scaling the images up by a fixed amount for all of the images does not cause a worsening of classification accuracy – a result that corroborates the conclusions of Dollar *et al.* (2014). This apparent contradiction is resolved in consideration of the fact that most of the images are not scaled up by the same factor and that the hypothesised effect of scaling the images up by different factors is to introduce scaling artefacts that degrade the quality of the features extracted due to the fixed methods of interpolation being applied to the images. This hypothesis is given some support by these results.

Depth Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8670	0.7257	0.7901
Pro-HOG (GSS, 72×72)	Linear	0.9011	0.6958	0.7853
Pro-HOG (72×72)	Linear	0.8839	0.7180	0.7924
Pro-HOG (GSS)	Linear	0.8960	0.7244	0.8011
Pro-HOG	Linear	0.9047	0.7453	0.8173
HOG	Non-Linear	0.9821	0.7470	0.8486
Pro-HOG (GSS, 72×72)	Non-Linear	0.9776	0.7065	0.8202
Pro-HOG (72×72)	Non-Linear	0.9790	0.7351	0.8397
Pro-HOG (GSS)	Non-Linear	0.9730	0.7223	0.8291
Pro-HOG	Non-Linear	0.9725	0.7389	0.8398

Table 4.7: Classification Results – AAM “Traffic Light” (Depth descriptors)

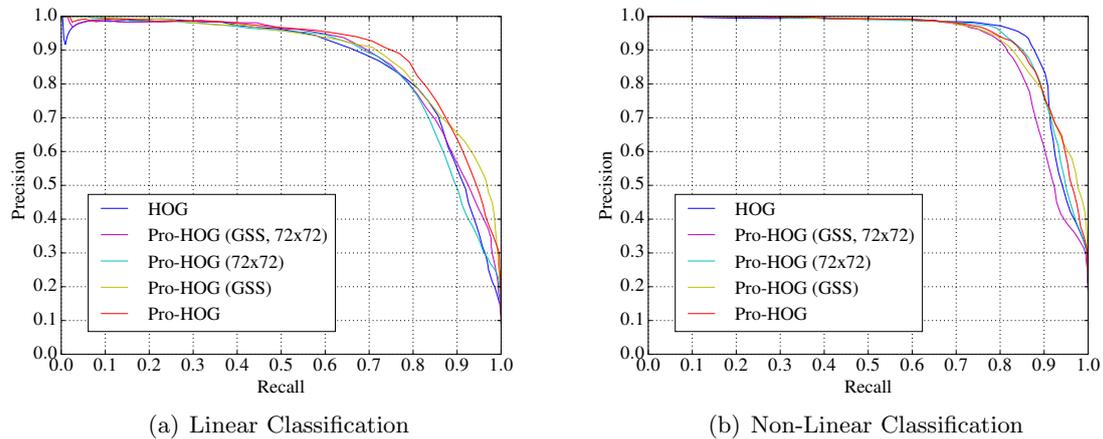


Figure 4-35: Precision versus Recall using descriptors extracted from depth images of the AAM “Traffic Light” object type.

Figure 4-35 shows that with the depth based descriptors, accuracy in precision and recall is improved over all feature extractor types compared to the intensity based descriptors. In the case of linear classification, the basic Pro-HOG variant (without GSS) gives the highest precision and recall accuracy. In the non-linear case, this variant of Pro-HOG is comparable to HOG in accuracy.

4.3.8 AAM “Triangular Road Sign”

Examples of the AAM “Triangular Road Sign” dataset are shown in figure 4-36. This dataset is similar in nature to the AAM “Traffic Light” dataset, but with a lower intraclass variance due to the fact that all of the sign posts are located directly underneath the triangular sign and the sign itself is always located in the top middle of the images. As in the case for most of the AAM object types, the examples are small and relatively indistinct. Figure 4-37 shows that most of the examples ($\approx 97\%$) have smaller dimensions than the fixed resizing dimensions meaning that resizing will cause most of the images to be scaled by factors > 1 . The variance in example sizes is low, meaning that the resizing factors will be similar and that any scaling artefacts introduced will be reasonably consistent and therefore representative of the examples.

While similar in nature to the AAM “Traffic Light” dataset, this dataset is smaller having only 163 examples as opposed to 1172. Notwithstanding the size of the dataset, it is expected that classification results will be similar to those for the AAM “Traffic Light” objects.



Figure 4-36: Sample extracts from the AAM “Triangular Road Sign” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

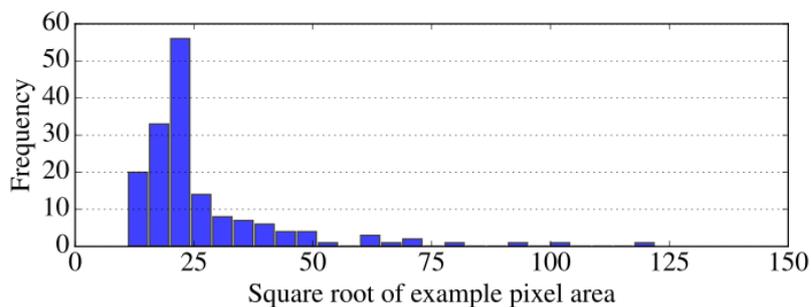


Figure 4-37: Distribution of example sizes for the AAM “Triangular Road Sign” object type.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9173	0.7822	0.8444
Pro-HOG (GSS, 72×72)	Linear	0.9030	0.8282	0.8640
Pro-HOG (72×72)	Linear	0.8605	0.7945	0.8262
Pro-HOG (GSS)	Linear	0.8930	0.8190	0.8544
Pro-HOG	Linear	0.8456	0.7393	0.7889
HOG	Non-Linear	0.9418	0.7945	0.8619
Pro-HOG (GSS, 72×72)	Non-Linear	0.9585	0.8497	0.9008
Pro-HOG (72×72)	Non-Linear	0.9606	0.8221	0.8860
Pro-HOG (GSS)	Non-Linear	0.9483	0.8436	0.8929
Pro-HOG	Non-Linear	0.9354	0.7546	0.8353

Table 4.8: Classification Results – AAM “Triangular Road Sign” (Intensity descriptors)

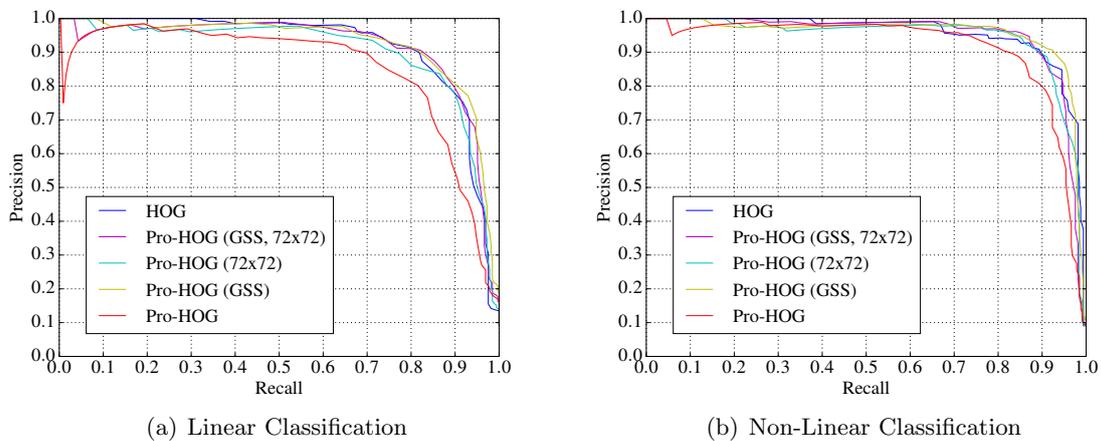


Figure 4-38: Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Triangular Road Sign” object type.

Intensity Based Descriptors

Figure 4-38 shows that the relative accuracy of the extractors is similar in both the linear and the non-linear classification cases, although accuracy in the non-linear case is higher overall. The expectation that the results would follow those of the AAM “Traffic Light” experiment is not borne out. In that experiment, both of the scale independent versions of Pro-HOG gave better results than either of the image resizing configurations or HOG. In this experiment, it is the choice of whether GSS is enabled that has a greater impact on accuracy. In the GSS disabled versions of Pro-HOG, resizing the images is a much better option than not. In both classification cases, the poorest performing extractor is Pro-HOG in its basic configuration.

With the configuration of Pro-HOG that is most similar to HOG (image resizing with GSS disabled), accuracy is approximately on a par with HOG. The two versions of Pro-HOG with GSS enabled are the best performing extractors but the results between those two cases are not significantly different (image resizing Pro-HOG is marginally better in both precision and recall). As in the case of the Pascal VOC “Car” dataset, the benefit of having GSS enabled or not is not independent of the choice of whether to resize the images to fixed dimensions prior to extracting the features; the behaviour of these two aspects of the algorithm, taken together with the object type under evaluation is influential in determining classification accuracy.

Depth Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9542	0.8957	0.9241
Pro-HOG (GSS, 72×72)	Linear	0.9479	0.8374	0.8893
Pro-HOG (72×72)	Linear	0.9621	0.8558	0.9058
Pro-HOG (GSS)	Linear	0.9452	0.8466	0.8932
Pro-HOG	Linear	0.9426	0.8558	0.8971
HOG	Non-Linear	0.9793	0.8712	0.9221
Pro-HOG (GSS, 72×72)	Non-Linear	0.9830	0.8865	0.9323
Pro-HOG (72×72)	Non-Linear	0.9930	0.8712	0.9281
Pro-HOG (GSS)	Non-Linear	0.9823	0.8497	0.9112
Pro-HOG	Non-Linear	0.9861	0.8712	0.9251

Table 4.9: Classification Results – AAM “Triangular Road Sign” (Depth descriptors)

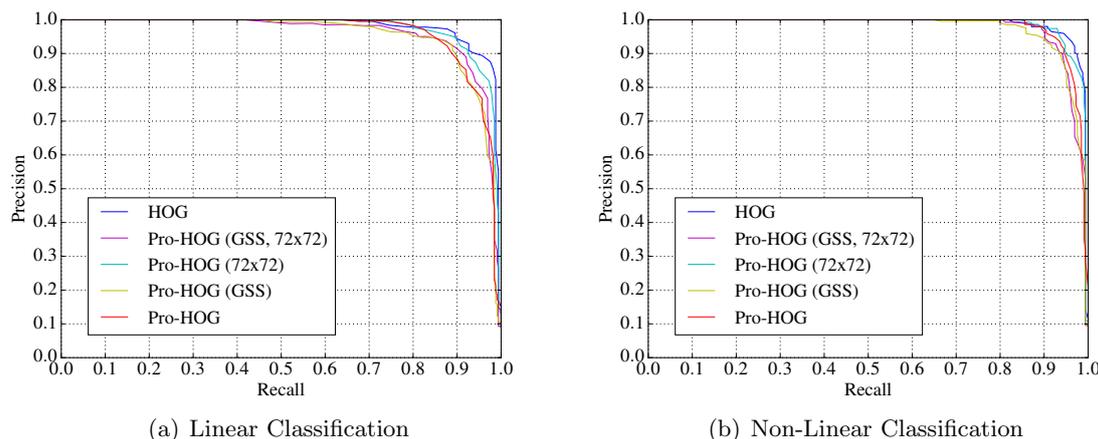


Figure 4-39: Precision versus Recall using feature descriptors extracted from depth images of the AAM “Triangular Road Sign” object type.

With the depth descriptors, classification accuracy is again much higher with this object

class than when using the intensity based descriptors in both the linear and the non-linear classification cases (as seen in figure 4-39). As in the AAM “Car” case, HOG again achieves slightly higher precision and recall over Pro-HOG. Given the relatively flat nature of this object type, the small changes in surface morphology which are not encoded by HOG (due to the previously explained rounding issues) may enhance the generalising ability of the classifiers over the Pro-HOG classifiers which are much more sensitive to small and sometimes spurious changes in depth.

4.3.9 AAM “Truck / Van”

Examples of the AAM “Truck / Van” class are given in figure 4-40. This class is more varied than the AAM “Car” class as it contains a less restricted object type. Some of the examples in this class might better be categorised as large cars or four-wheel drive vehicles. The relative size of the objects leads to much more variety in background too though all of the vehicles are still predominantly located on grey homogeneous road surfaces.

Figure 4-41 shows that there is more variety in the pixel dimensions of the examples in this dataset compared to the previous AAM datasets. Based on the results of the previous experiments, the greater variation in appearance and example dimensions points to Pro-HOG in its original configuration (either GSS enabled or not) giving improved classification accuracy.

Intensity Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8528	0.6782	0.7556
Pro-HOG (GSS, 72×72)	Linear	0.9169	0.7633	0.8331
Pro-HOG (72×72)	Linear	0.8431	0.6862	0.7566
Pro-HOG (GSS)	Linear	0.9097	0.7766	0.8379
Pro-HOG	Linear	0.8424	0.6968	0.7627
HOG	Non-Linear	0.9470	0.6649	0.7812
Pro-HOG (GSS, 72×72)	Non-Linear	0.9675	0.7926	0.8713
Pro-HOG (72×72)	Non-Linear	0.9304	0.6755	0.7827
Pro-HOG (GSS)	Non-Linear	0.9325	0.7713	0.8443
Pro-HOG	Non-Linear	0.9375	0.7181	0.8133

Table 4.10: Classification Results – AAM “Truck / Van” (Intensity descriptors)

In the linear case, figure 4-42(a) shows that having GSS enabled is the most important



Figure 4-40: Sample extracts from the AAM “Truck / Van” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

factor in achieving high classification accuracy for this object type. Both the GSS disabled versions of Pro-HOG give classification accuracy that is no better than HOG. The results in the non-linear case shown in figure 4-42(b), again show improved accuracy across all of the extractors, but with the best method being the image resizing version of Pro-HOG. This is counter-intuitive given the results so far and the expectations that classifying an object type having larger variability in the size of its examples would benefit more from the scale-independent nature of Pro-HOG in its non-resizing configuration. HOG results in the lowest classification accuracy in both the linear and the non-linear case. The version of Pro-HOG that is algorithmically most similar to HOG (with image resizing and GSS disabled) has almost exactly the same accuracy as HOG – with better recall but slightly worse precision.

Depth Based Descriptors

With the depth based features, the differences between the different feature extractors become less pronounced. Again, the overall accuracy in terms of both precision and recall is improved over the intensity based descriptors, but GSS is no longer helping to improve accuracy for the two Pro-HOG methods that implement it. Overall, HOG performs better than Pro-

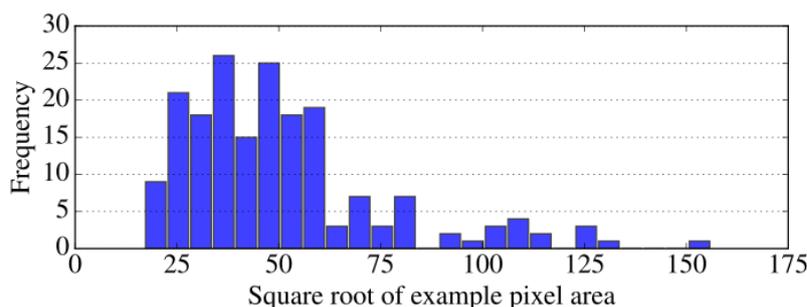


Figure 4-41: Distribution of example sizes for the AAM “Truck / Van” object type.

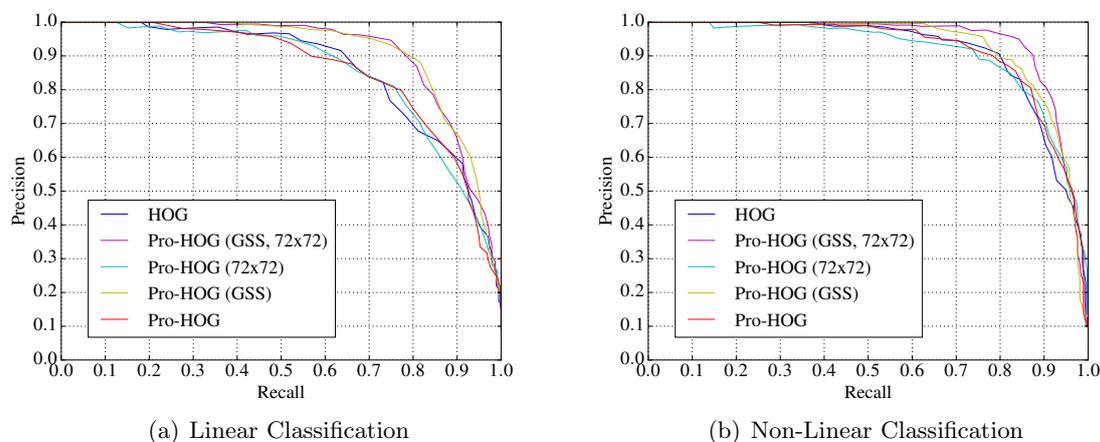


Figure 4-42: Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Truck / Van” object type.

HOG with the depth based descriptors. Most tellingly, while the version of Pro-HOG that is algorithmically closest to HOG performed almost identically to HOG with the intensity based descriptors, with the depth descriptors it is slightly (though significantly) less accurate in precision and recall. Given that most of the examples of trucks and vans are in profile view and presenting a strong planar silhouette, the reason for this difference can most likely be attributed to the depth rounding undertaken by HOG (the same reason as in the previous case with the AAM “Triangular Road Sign” object type).

4.3.10 AAM “Road Light”

A selection of examples from the AAM “Road Light” class are shown in figure 4-44. This object type expresses the greatest appearance variability within the AAM dataset tested so far. It is also one of the largest datasets tested which gives greater confidence in the interpretation

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8988	0.8032	0.8483
Pro-HOG (GSS, 72×72)	Linear	0.8958	0.8005	0.8455
Pro-HOG (72×72)	Linear	0.8889	0.7660	0.8229
Pro-HOG (GSS)	Linear	0.8770	0.7394	0.8023
Pro-HOG	Linear	0.8665	0.7420	0.7994
HOG	Non-Linear	0.9505	0.8165	0.8784
Pro-HOG (GSS, 72×72)	Non-Linear	0.9669	0.7766	0.8614
Pro-HOG (72×72)	Non-Linear	0.9487	0.7872	0.8605
Pro-HOG (GSS)	Non-Linear	0.9656	0.7473	0.8426
Pro-HOG	Non-Linear	0.9281	0.7553	0.8328

Table 4.11: Classification Results – AAM “Truck / Van” (Depth descriptors)

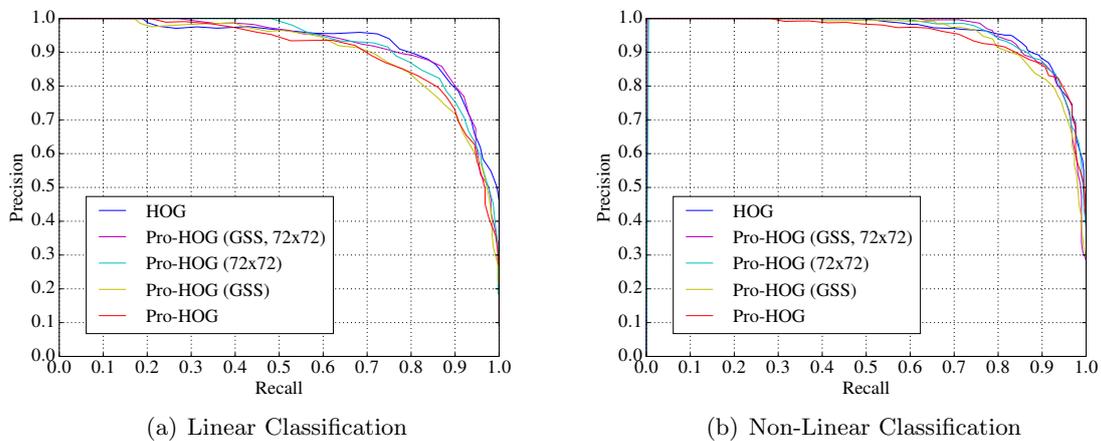


Figure 4-43: Precision versus Recall using feature descriptors extracted from depth images of the AAM “Truck / Van” object type.

of results attached to this class, if there are significant differences in classification accuracy between the extractor types. Figure 4-45 shows the distribution of image sizes in this class. Only $\approx 26\%$ of the images are below the fixed resize dimensions, meaning that the vast majority of the examples will be rescaled down, possibly losing information. However, given that the salient characteristics of this object type are strongly contrasting vertical poles, it is unlikely that downsizing the images will result in information loss that is significant enough to impact the quality of feature encoding and consequent classification accuracy.

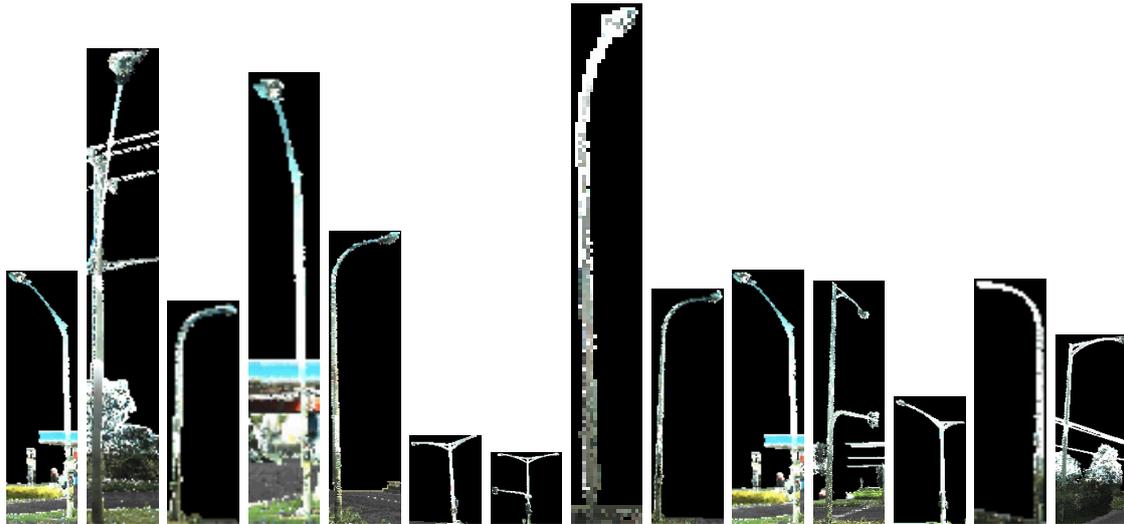


Figure 4-44: Sample extracts from the AAM “Road Light” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

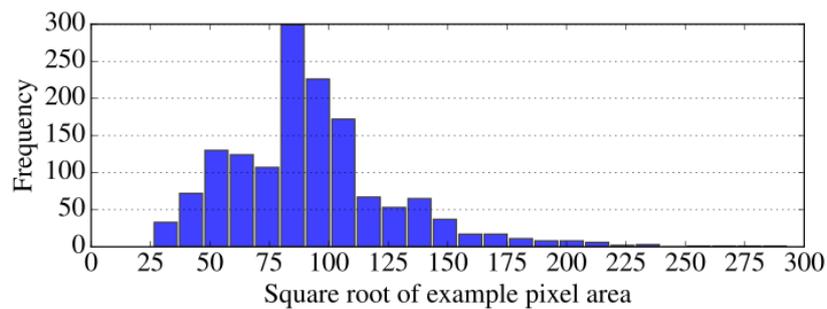


Figure 4-45: Distribution of example sizes for the AAM “Road Light” object type.

Intensity Based Descriptors

For this object type, classification accuracy is high in both the linear and the non-linear cases of classification. Since the object extracts are almost always present with the sky as background, this accuracy is probably due to the strong intensity gradient between the objects and the sky’s black background (which has no depth information and so no colour information is mapped either).

In the linear classification results shown in figure 4-46(a), the most important attribute of the Pro-HOG extractors for achieving good accuracy is the use of GSS. The two GSS disabled versions of Pro-HOG (that resize and that do not resize the images) achieve similar levels of precision and recall and are both better than HOG, but not as good as the GSS enabled versions of Pro-HOG. For this object type, the data appear to be distinctive enough for the

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9317	0.8631	0.8961
Pro-HOG (GSS, 72×72)	Linear	0.9644	0.9073	0.9349
Pro-HOG (72×72)	Linear	0.9467	0.8932	0.9192
Pro-HOG (GSS)	Linear	0.9603	0.9182	0.9388
Pro-HOG	Linear	0.9424	0.8901	0.9155
HOG	Non-Linear	0.9823	0.9127	0.9462
Pro-HOG (GSS, 72×72)	Non-Linear	0.9831	0.9144	0.9475
Pro-HOG (72×72)	Non-Linear	0.9805	0.9117	0.9448
Pro-HOG (GSS)	Non-Linear	0.9809	0.9144	0.9465
Pro-HOG	Non-Linear	0.9790	0.9093	0.9429

Table 4.12: Classification Results – AAM “Road Light” (Intensity descriptors)

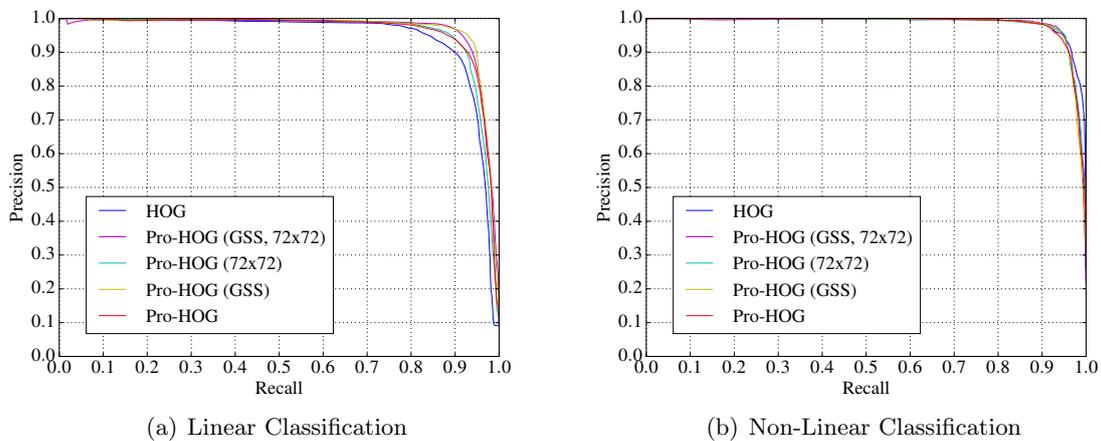


Figure 4-46: Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Road Light” object type.

quality of feature encoding to not be degraded by the resizing of the extracts. In the results for the non-linear case shown in figure 4-46(b), classification accuracy across all five of the extractors is nearly identical.

Depth Based Descriptors

In the case of the depth descriptors, classification accuracy using Pro-HOG is nearly identical to the accuracy attained with the intensity based descriptors – in both the linear and the non-linear cases of classification (as shown in figure 4-47). The GSS enabled versions of Pro-HOG still give better accuracy than the non GSS enabled versions of Pro-HOG. In the case of HOG, whereas accuracy was worse with the intensity based descriptors, accuracy is improved

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9447	0.8901	0.9167
Pro-HOG (GSS, 72×72)	Linear	0.9568	0.9168	0.9364
Pro-HOG (72×72)	Linear	0.9376	0.8850	0.9106
Pro-HOG (GSS)	Linear	0.9610	0.9185	0.9393
Pro-HOG	Linear	0.9525	0.8925	0.9216
HOG	Non-Linear	0.9843	0.9251	0.9538
Pro-HOG (GSS, 72×72)	Non-Linear	0.9839	0.9227	0.9523
Pro-HOG (72×72)	Non-Linear	0.9853	0.9196	0.9513
Pro-HOG (GSS)	Non-Linear	0.9825	0.9220	0.9513
Pro-HOG	Non-Linear	0.9824	0.9158	0.9479

Table 4.13: Classification Results – AAM “Road Light” (Depth descriptors)

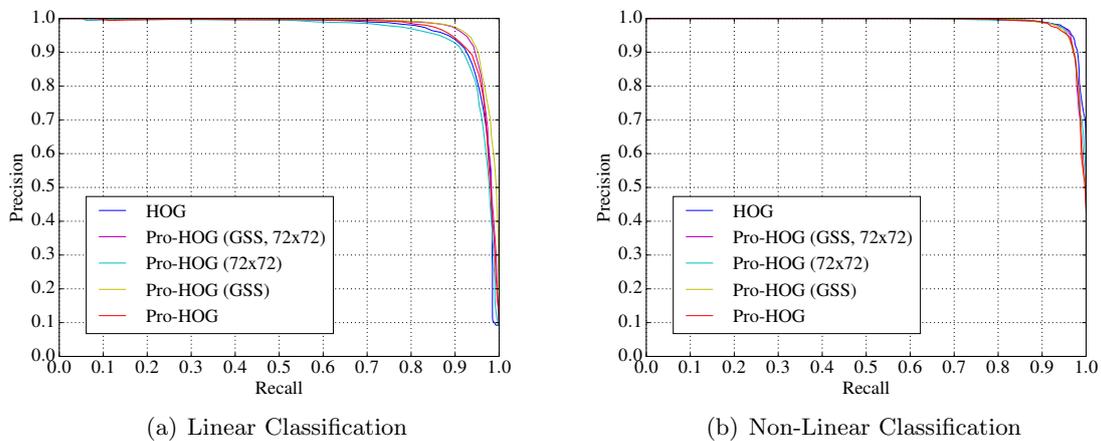


Figure 4-47: Precision versus Recall using feature descriptors extracted from depth images of the AAM “Road Light” object type.

to the same level of Pro-HOG (in its base configuration) using the depth based descriptors. Pro-HOG in its HOG emulating configuration (image resizing and GSS disabled), is slightly less accurate in precision and recall over all classification thresholds compared to HOG.

4.3.11 AAM “Telegraph Pole”

In qualitative terms, the AAM “Telegraph Pole” object type is similar to the AAM “Road Light” class in its level of intraclass appearance variability. Like the AAM “Road Light” class, it has a dominant structure in the form of a long vertical pole – this time wider and with horizontal cross pieces. It is also a reasonably large dataset (937 examples). However, the distribution of the image sizes is tighter than in the AAM “Road Light” dataset, with many more examples smaller than the fixed resize dimensions (as seen in figure 4-49). This makes the essential point of difference between this object type and the AAM “Road Light” object type to be the dimensions of the examples. In this experiment, more of the examples will be up scaled than not, and by different scaling factors > 1 (there is a reasonably even spread of example sizes beneath the 72×72 resize point).



Figure 4-48: Sample extracts from the AAM “Telegraph Pole” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

Intensity Based Descriptors

Figure 4-50(a) shows that in the linear classification case, Pro-HOG has greater overall accuracy in all of its configurations than HOG, and in the non-linear case, figure 4-50(b) shows that there is little significant difference between any of the feature extractor types. In the

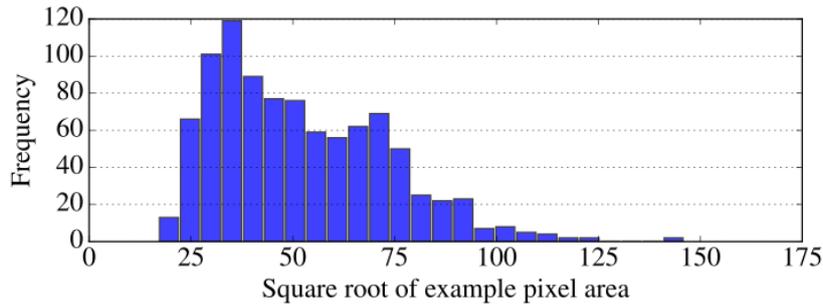


Figure 4-49: Distribution of example sizes for the AAM “Telegraph Pole” object type.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9220	0.8324	0.8749
Pro-HOG (GSS, 72×72)	Linear	0.9271	0.8757	0.9007
Pro-HOG (72×72)	Linear	0.9268	0.8719	0.8985
Pro-HOG (GSS)	Linear	0.9282	0.8490	0.8868
Pro-HOG	Linear	0.9173	0.8639	0.8898
HOG	Non-Linear	0.9479	0.8543	0.8987
Pro-HOG (GSS, 72×72)	Non-Linear	0.9484	0.8634	0.9039
Pro-HOG (72×72)	Non-Linear	0.9491	0.8762	0.9112
Pro-HOG (GSS)	Non-Linear	0.9547	0.8650	0.9076
Pro-HOG	Non-Linear	0.9485	0.8549	0.8992

Table 4.14: Classification Results – AAM “Telegraph Pole” (Intensity descriptors)

more distinctive linear case, the reason for the separation between the different Pro-HOG classes is not the same as in the AAM “Road Light” class. For the AAM “Road Light” class, enabling GSS allows for improved accuracy. For this object type, Pro-HOG gives improved recall when resizing the extracts to fixed dimensions. Enabling GSS improves precision in the Pro-HOG configurations, but the effect is very small (as seen at the default threshold level in table 4.14).

In its most similar configuration to HOG, Pro-HOG gives significantly better recall (but with a similar level of precision to HOG). Even though Pro-HOG lacks some of the features of the HOG algorithm (*i.e.* cell resolution Gaussian filtering and cell histogram interpolation), this results in improved rather than impaired classification accuracy for this object type. One possible reason for this could be that with HOG, applying cell resolution Gaussian filtering on top of the already “smoothed” enlarged examples (due to the extracts mainly being of an original size that is smaller than the image resizing dimensions – the scaling up acting to interpolate the image pixels), causes too much blurring around the object boundaries and could be degrading the quality of the extracted features.

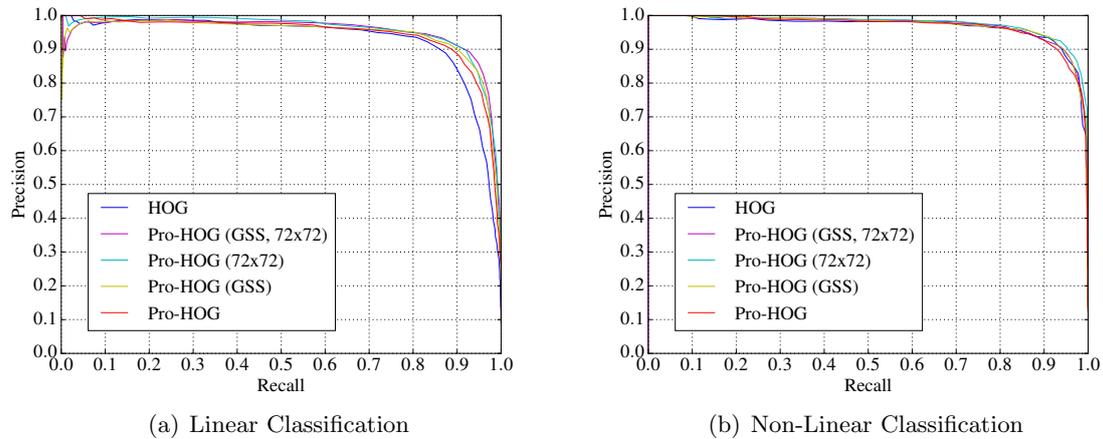


Figure 4-50: Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Telegraph Pole” object type.

The telegraph poles have a lot of thin protrusions that are especially indicative of the object type; too much blurring might diminish the appearance of these parts. In Pro-HOG, the lack of cell grid resolution Gaussian filtering (Gaussian filtering is applied using a fixed 3×3 pixel kernel instead) leaves objects with more distinctive boundaries so the encoded features can be more representative.

In the case of non-linear classification, the fact that the descriptors might be slightly less representative of the objects using HOG has less of an effect on classification accuracy because the non-linear boundary is less affected by such slight inaccuracies in feature representation allowing HOG to perform at the same level of accuracy as Pro-HOG.

Depth Based Descriptors

With the depth based descriptors, figure 4-51(a) shows that HOG no longer results in lower classification accuracy than Pro-HOG in the case of linear classification, and is comparable with the accuracy achieved using the resizing versions of Pro-HOG. It is likely that the more definite contrast gradients present in depth means that the more aggressive blurring in HOG’s algorithm does not degrade the object representations (as given by the feature descriptors) as much as when deriving object descriptors from the intensity images. In addition, the integer rounding of depth used by HOG may help to standardise the appearance (in depth) of the telegraph poles when present in the images at an oblique angle.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9437	0.8853	0.9135
Pro-HOG (GSS, 72×72)	Linear	0.9462	0.8911	0.9178
Pro-HOG (72×72)	Linear	0.9378	0.8767	0.9062
Pro-HOG (GSS)	Linear	0.9250	0.8751	0.8994
Pro-HOG	Linear	0.9264	0.8794	0.9023
HOG	Non-Linear	0.9566	0.9173	0.9365
Pro-HOG (GSS, 72×72)	Non-Linear	0.9643	0.8933	0.9274
Pro-HOG (72×72)	Non-Linear	0.9618	0.8991	0.9294
Pro-HOG (GSS)	Non-Linear	0.9589	0.9077	0.9326
Pro-HOG	Non-Linear	0.9620	0.8911	0.9252

Table 4.15: Classification Results – AAM “Telegraph Pole” (Depth descriptors)

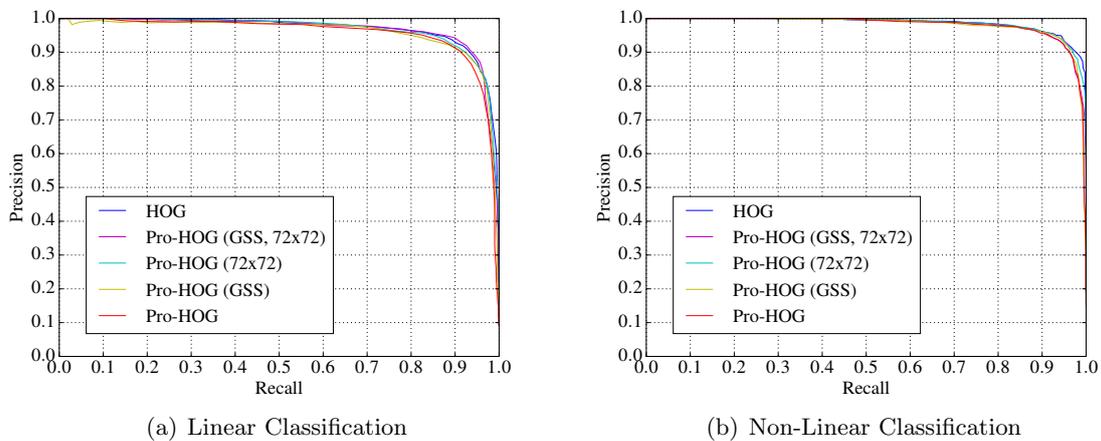


Figure 4-51: Precision versus Recall using feature descriptors extracted from depth images of the AAM “Telegraph Pole” object type.

4.3.12 AAM “Rectangular Road Sign”

Some examples from the AAM “Rectangular Road Sign” dataset are shown in figure 4-52. The extracts tend to be represented by one or two vertical poles in the lower half of the image with the rectangular sign itself taking up the top most part of the image. Although the dataset contains a large number of examples (668), most of the examples are small and indistinct. As seen in figure 4-53, 97% of the examples are smaller than the fixed resize dimensions, with most of the examples having dimensions around 24×39 pixels.

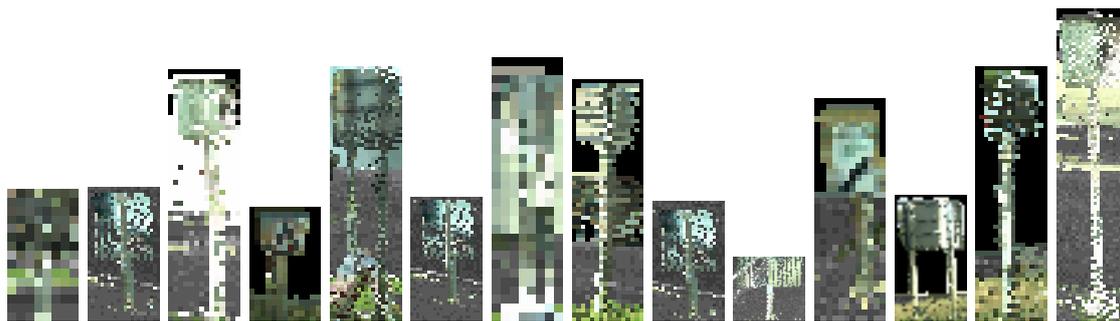


Figure 4-52: Sample extracts from the AAM “Rectangular Road Sign” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

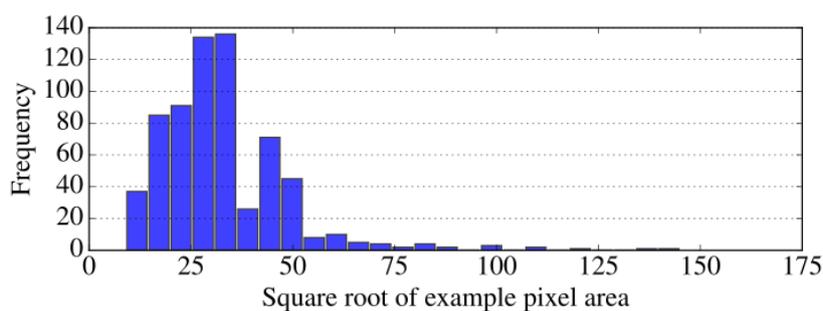


Figure 4-53: Distribution of example sizes for the AAM “Rectangular Road Sign” object type.

Intensity Based Descriptors

In the linear classification case, the results for the AAM “Rectangular Road Sign” object class (see figure 4-54(a)) are similar to those for the AAM “Telegraph Pole” class (see figure 4-50(a)) but with lower levels of accuracy overall, and more variability in the results because of the smaller size of the dataset; the overall ordering of the different feature extractors in the results is similar. As in the AAM “Telegraph Pole” case, the image resizing versions of Pro-HOG give the greatest accuracy. Enabling GSS improves accuracy mainly in precision, but only in Pro-HOG’s non-resizing configuration. Enabling GSS in Pro-HOG’s image resizing configuration decreases recall and only increases precision marginally. Figure 4-54(b) shows that in the non-linear classification case, there is little difference between all of the extractor types with overall accuracy higher than in the linear case.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8733	0.7897	0.8294
Pro-HOG (GSS, 72×72)	Linear	0.9023	0.8016	0.8490
Pro-HOG (72×72)	Linear	0.8960	0.8189	0.8557
Pro-HOG (GSS)	Linear	0.8940	0.7762	0.8309
Pro-HOG	Linear	0.8482	0.7695	0.8069
HOG	Non-Linear	0.9393	0.7522	0.8354
Pro-HOG (GSS, 72×72)	Non-Linear	0.9466	0.7695	0.8489
Pro-HOG (72×72)	Non-Linear	0.9388	0.7919	0.8591
Pro-HOG (GSS)	Non-Linear	0.9332	0.7627	0.8394
Pro-HOG	Non-Linear	0.9310	0.7575	0.8353

Table 4.16: Classification Results – AAM “Rectangular Road Sign” (Intensity descriptors)

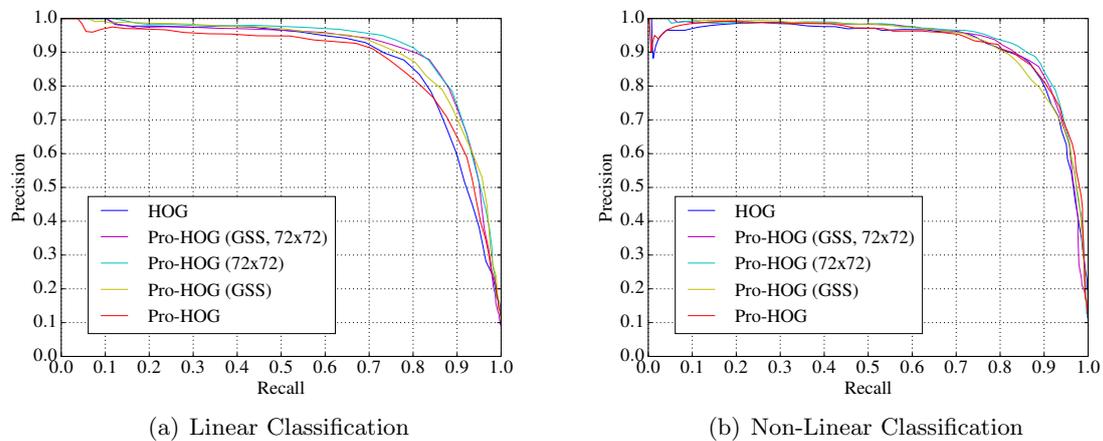


Figure 4-54: Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Rectangular Road Sign” object type.

Depth Based Descriptors

As in the previous examples, figure 4-55 shows that the depth descriptors give improved accuracy over the intensity based descriptors for all extractor configurations. Again, HOG has notably better performance using the depth based descriptors and is comparable in accuracy to the Pro-HOG extractors. Unusually, the Pro-HOG configuration that is most like HOG (image resizing and GSS disabled) provides for significantly less accuracy than any of the other Pro-HOG configurations. The best performing configuration of Pro-HOG is also the least similar to the HOG extractor. This is the non-resizing and GSS enabled variant. In the experiments with the intensity based descriptors, this version of Pro-HOG was the better of the two non-resizing variants, but still performed less well than both of the image resizing variants. This indicates that image resizing serves as a positive (or non-inhibiting) influence

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9351	0.8308	0.8799
Pro-HOG (GSS, 72×72)	Linear	0.9276	0.7957	0.8566
Pro-HOG (72×72)	Linear	0.9367	0.8308	0.8806
Pro-HOG (GSS)	Linear	0.9385	0.8451	0.8893
Pro-HOG	Linear	0.9256	0.8293	0.8749
HOG	Non-Linear	0.9701	0.8503	0.9063
Pro-HOG (GSS, 72×72)	Non-Linear	0.9667	0.8473	0.9031
Pro-HOG (72×72)	Non-Linear	0.9692	0.8466	0.9037
Pro-HOG (GSS)	Non-Linear	0.9666	0.8443	0.9013
Pro-HOG	Non-Linear	0.9689	0.8383	0.8989

Table 4.17: Classification Results – AAM “Rectangular Road Sign” (Depth descriptors)

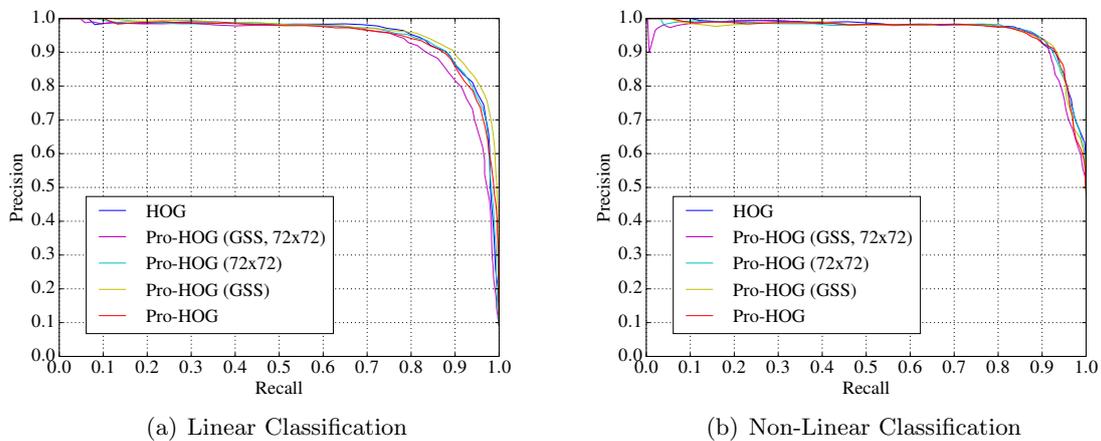


Figure 4-55: Precision versus Recall using feature descriptors extracted from depth images of the AAM “Rectangular Road Sign” object type.

when using the intensity based descriptors, but that (in the case of this particular object type) it degrades the accuracy of the depth based descriptors – and only in the case of Pro-HOG. Since HOG resizes the images but does not suffer in accuracy to the same degree, it must be that HOG’s more involved cell histogram interpolation and cell resolution Gaussian smoothing methods help to mitigate feature descriptor degradation in this case.

4.3.13 AAM “Generic Road Sign”

The AAM “Generic Road Sign” class is the final of the AAM object types and it has much greater variation in appearance than either of the other two AAM road sign classes. This class was used for examples that couldn’t be more properly labelled as instances of either

the rectangular or the triangular road sign classes. Some examples of the type are shown in figure 4-56. As seen in figure 4-57, the examples in this class vary similarly in their image dimensions to the AAM “Rectangular Road Sign” class although the dataset is more than twice as large at 1489 examples.



Figure 4-56: Sample extracts from the AAM “Generic Road Sign” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

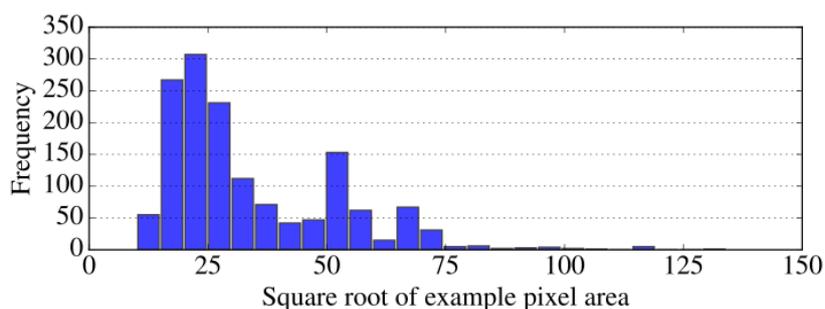


Figure 4-57: Distribution of example sizes for the AAM “Generic Road Sign” object type.

Intensity Based Descriptors

In the case of linear classification, figure 4-58(a) shows that Pro-HOG in all of its configurations gives significantly improved accuracy over HOG, with the GSS enabled, non-resizing version of Pro-HOG resulting in the highest precision and recall classification scores. As seen in figure 4-58(b), in the case of non-linear classification, there are no significant differences between any of the feature extractor types. At the default threshold in the non-linear case shown in table 4.18, Pro-HOG gives better recall than HOG by one or two percentage points at most while precision is approximately unchanged across the different feature extractors. In the linear case, even the Pro-HOG configuration that is most like HOG gives far better accuracy. This indicates that for this object type, the aspects of the HOG algorithm that cannot be implemented in Pro-HOG are not hindering Pro-HOG, but are in fact helping Pro-HOG to derive descriptors that are better suited for classification.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8491	0.7330	0.7868
Pro-HOG (GSS, 72×72)	Linear	0.9033	0.7837	0.8393
Pro-HOG (72×72)	Linear	0.8898	0.7861	0.8347
Pro-HOG (GSS)	Linear	0.9074	0.7999	0.8503
Pro-HOG	Linear	0.8818	0.7888	0.8327
HOG	Non-Linear	0.9417	0.8026	0.8666
Pro-HOG (GSS, 72×72)	Non-Linear	0.9493	0.8167	0.8780
Pro-HOG (72×72)	Non-Linear	0.9500	0.8230	0.8820
Pro-HOG (GSS)	Non-Linear	0.9403	0.8146	0.8730
Pro-HOG	Non-Linear	0.9437	0.8113	0.8725

Table 4.18: Classification Results – AAM “Generic Road Sign” (Intensity descriptors)

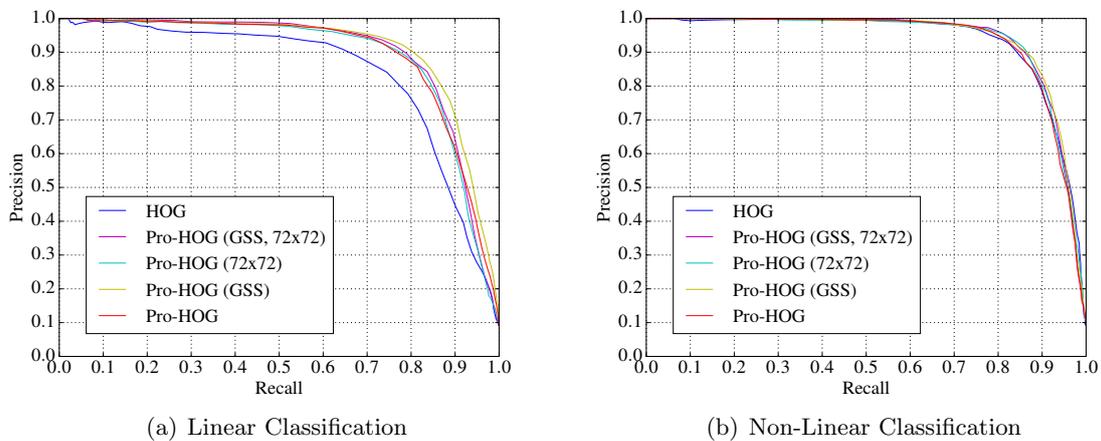


Figure 4-58: Precision versus Recall using feature descriptors extracted from intensity images of the AAM “Generic Road Sign” object type.

Depth Based Descriptors

Figure 4-59 shows that the depth based descriptors again demonstrate increased accuracy over the intensity based descriptors across all of the extractor configurations. In the linear case, figure 4-59(a) shows that HOG still gives lower classification accuracy than any of the Pro-HOG extractor configurations, and that the GSS enabled, non-resizing version of Pro-HOG is again the best performing of the four different Pro-HOG configurations although the relative increase in improvement is much smaller than when using the intensity based descriptors. The results for this object type are unusual in that this is the only case (in the AAM dataset) where the use of depth based descriptors do not allow HOG to perform at a comparable level of accuracy to the Pro-HOG extractors. For the intensity based descriptors, figure 4-58(a) shows that in the linear case, all of the extractors have a strong bias towards

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9035	0.8610	0.8817
Pro-HOG (GSS, 72×72)	Linear	0.9329	0.8734	0.9022
Pro-HOG (72×72)	Linear	0.9159	0.8744	0.8947
Pro-HOG (GSS)	Linear	0.9278	0.8852	0.9060
Pro-HOG	Linear	0.9208	0.8707	0.8951
HOG	Non-Linear	0.9713	0.8983	0.9334
Pro-HOG (GSS, 72×72)	Non-Linear	0.9716	0.8952	0.9318
Pro-HOG (72×72)	Non-Linear	0.9678	0.8895	0.9270
Pro-HOG (GSS)	Non-Linear	0.9695	0.9070	0.9372
Pro-HOG	Non-Linear	0.9715	0.8932	0.9307

Table 4.19: Classification Results – AAM “Generic Road Sign” (Depth descriptors)

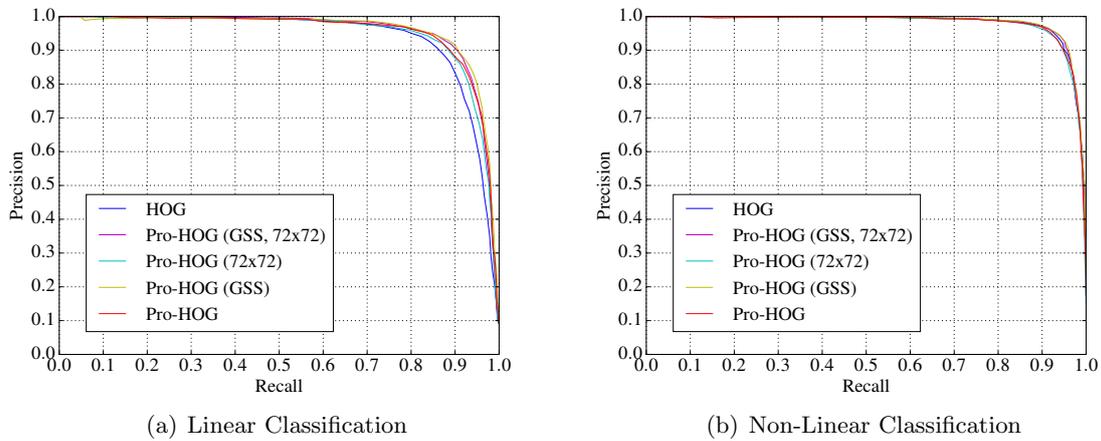


Figure 4-59: Precision versus Recall using feature descriptors extracted from depth images of the AAM “Generic Road Sign” object type.

precision over recall (indicated by a greater proportion of the mass under the curves being towards the top left corner of the graph). In the case of the linear depth based descriptors, this bias is much less pronounced – the masses under the curves are more evenly distributed on either side of the bottom left to top right diagonal.

4.3.14 Earthmine “Car”

There are fewer examples per object type in the Earthmine dataset due to the difficulties in acquiring accurate depth information from stereoscopic imaging. This means that less can be inferred from the results of the experiments on the Earthmine object types.

As in the AAM dataset, because of the manner in which the Earthmine data were collected, the viewpoints — and thus the orientations of the objects — are more constrained. The extracts of cars in this dataset appear in more restricted orientations (profile and front/back views) than those in the Pascal VOC dataset. However, unlike the AAM dataset where colour (and thus intensity) information was poorly mapped, the examples in the Earthmine dataset have colour information of a similar quality to the Pascal VOC dataset. The reduced intraclass variance due to the objects having more restricted orientations should result in improved classification accuracy scores even though there are only 52 examples in this object category.

Some examples of the Earthmine “Car” dataset are shown in figure 4-60. As seen in figure 4-61, the range of example sizes is much greater than in the AAM “Car” dataset, although not as great as in the Pascal VOC “Car” dataset. Unlike the AAM “Car” dataset, the sizes of the extracts in the Earthmine “Car” dataset are generally larger since they are much closer to the camera. This means that only 22 of the 52 images ($\approx 42\%$) are below the fixed image resize dimensions.

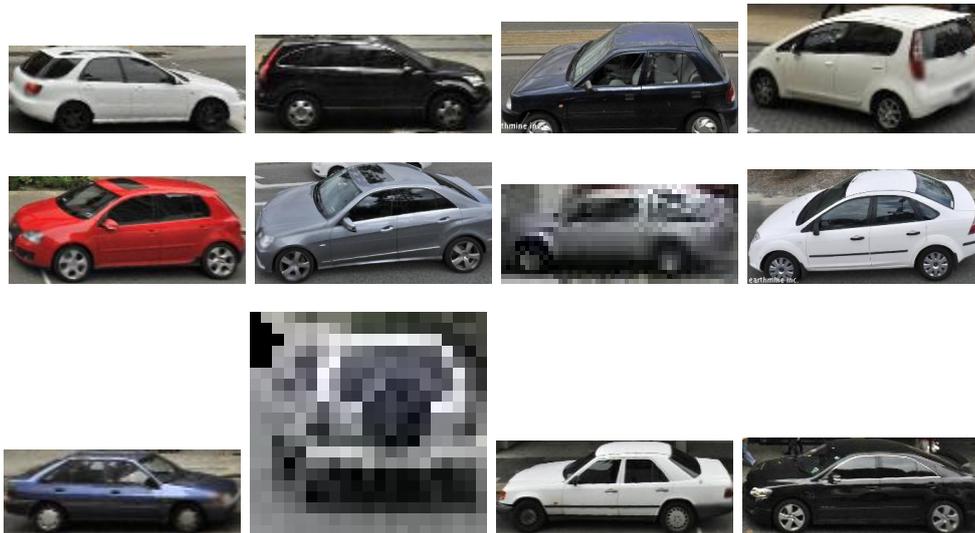


Figure 4-60: Sample extracts from the Earthmine “Car” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

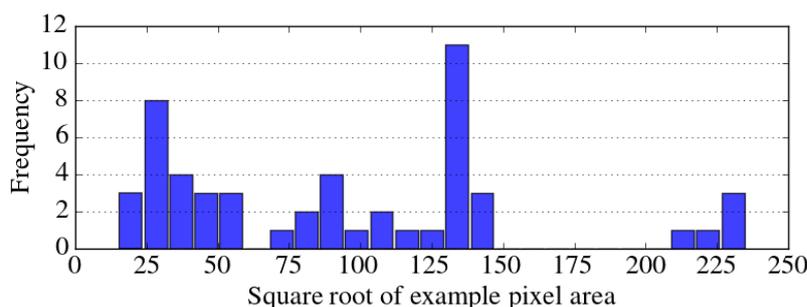


Figure 4-61: Distribution of example sizes for the Earthmine “Car” object type.

Intensity Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9778	0.8462	0.9072
Pro-HOG (GSS, 72×72)	Linear	1.0000	0.9327	0.9652
Pro-HOG (72×72)	Linear	0.9706	0.9519	0.9612
Pro-HOG (GSS)	Linear	0.9796	0.9231	0.9505
Pro-HOG	Linear	0.9787	0.8846	0.9293
HOG	Non-Linear	1.0000	0.8365	0.9110
Pro-HOG (GSS, 72×72)	Non-Linear	1.0000	0.8846	0.9388
Pro-HOG (72×72)	Non-Linear	1.0000	0.9423	0.9703
Pro-HOG (GSS)	Non-Linear	0.9894	0.8942	0.9394
Pro-HOG	Non-Linear	1.0000	0.8750	0.9333

Table 4.20: Classification Results – Earthmine “Car” (Intensity descriptors)

The small size of the Earthmine “Car” dataset means that the precision-recall graphs shown in figure 4-62 are not very helpful in identifying which of the extractors perform better, although it appears clear that the non-linear case provides for improved accuracy over the linear classification case.

At the default classification threshold shown in table 4.20, in the linear case, Pro-HOG in all four of its configurations exceeds HOG’s recall accuracy by between 4 and 11 percentage points, and Pro-HOG has comparable precision to HOG. Enabling GSS is preferred for both the resizing and non-resizing versions of Pro-HOG. In the non-linear classification case, precision is perfect in all but one of the extractor configurations. This indicates a low false-positive rate and that the classifiers have overfit to the training data – *i.e.* they are not generalising well enough. This is borne out by the relatively much lower (by an average of 10 percentage points) recall scores. This can happen if within the object type there are specific sub-types (in this case, colours or models of cars) that are more prevalent than others. This will cause certain features to be weighted more heavily than others by the machine learning algorithm,

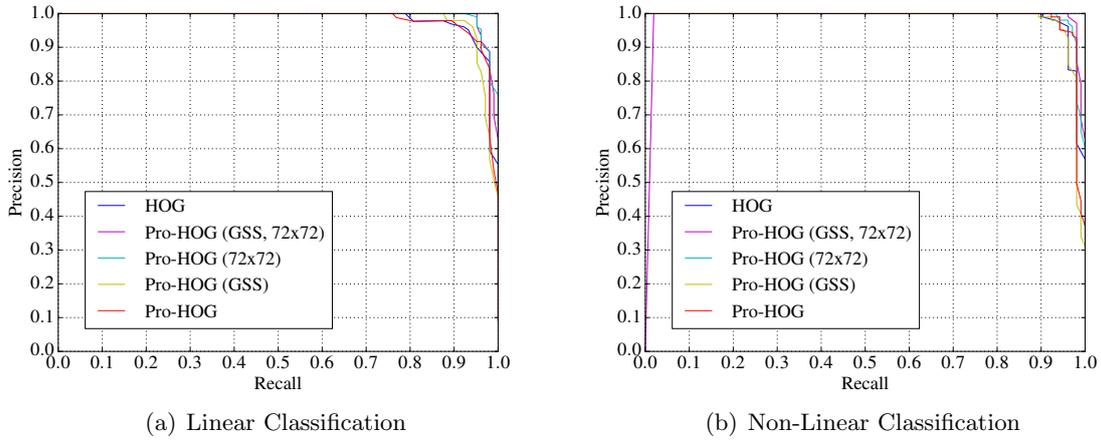


Figure 4-62: Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Car” object type.

resulting in a biased classifier.

Depth Based Descriptors

In all of the previous experiments with the AAM object types, the use of depth descriptors gives improved classification performance over the use of descriptors extracted from the intensity data. However, using depth descriptors with the Earthmine dataset results in lower classification accuracy than when using the intensity based descriptors.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9778	0.8462	0.9072
Pro-HOG (GSS, 72×72)	Linear	0.9556	0.8269	0.8866
Pro-HOG (72×72)	Linear	0.9022	0.7981	0.8469
Pro-HOG (GSS)	Linear	0.8333	0.7692	0.8000
Pro-HOG	Linear	0.8851	0.7404	0.8063
HOG	Non-Linear	1.0000	0.7115	0.8315
Pro-HOG (GSS, 72×72)	Non-Linear	0.9500	0.7308	0.8261
Pro-HOG (72×72)	Non-Linear	1.0000	0.6827	0.8114
Pro-HOG (GSS)	Non-Linear	0.9286	0.7500	0.8298
Pro-HOG	Non-Linear	0.9851	0.6346	0.7719

Table 4.21: Classification Results – Earthmine “Car” (Depth descriptors)

With the AAM data, the depth data generated by laser scanning is comparatively of a much higher quality than its colour / intensity data. This situation is reversed with the Earthmine dataset. The depth data generated by stereoscopy in the Earthmine data is much less accurate

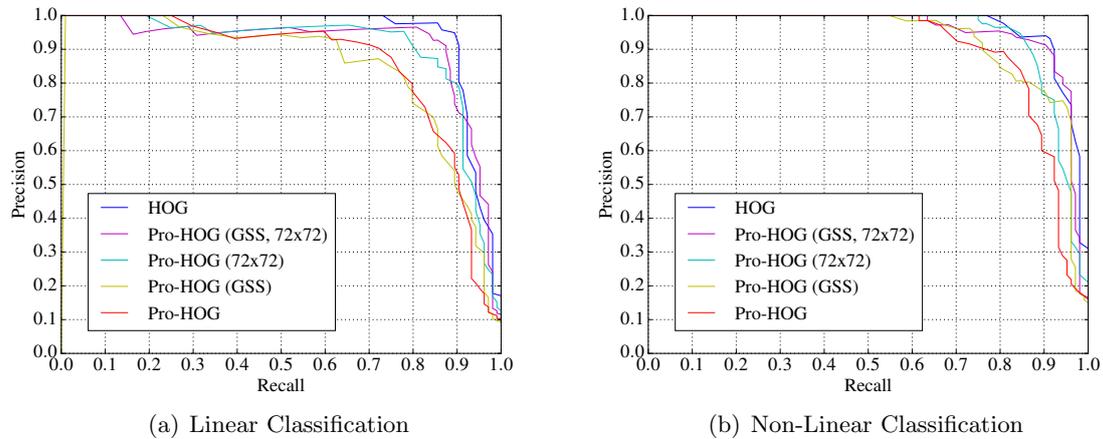


Figure 4-63: Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Car” object type.

than the laser scan generated depth data in the AAM dataset, but the 2-D colour information is of a much higher quality. With the Earthmine data, this means that the surface morphology of the objects is likely to be noisy and inaccurate and not reliable enough to be usefully encoded by the feature extractors. Moreover, the silhouettes of objects as described in depth are often poorly represented meaning that even the object boundaries cannot be accurately encoded in the descriptors.

The HOG implementation being used in these experiments rounds depth data to the nearest metre. Poor quality depth information will therefore be more of a problem for the Pro-HOG extractors which retain the input data’s original floating point values. The Pro-HOG extractors are more likely to encode spurious variations in depth that are not truly representative of the objects. This is seen in figure 4-63 where the Pro-HOG extractors do not perform as well as the HOG extractor. The two better performing versions of Pro-HOG (in both the linear and the non-linear classification cases) are the extract resizing versions. Since the bulk of the extracts for this object type have original dimensions that are larger than the resizing dimensions, decreasing the size of the extracts has the effect of reducing the impact of local noisy “bumps” over the surface of the objects meaning that the generated feature descriptors are less influenced by random noise and are more representative of the object’s type.

4.3.15 Earthmine “Garbage Bin”

The Earthmine “Garbage Bin” dataset expresses extremely little variation in appearance. The objects themselves are nearly identical and are typically placed against a flat grey background of paving stones. The bins themselves are all dark grey, cylindrical and supported by four slim upright posts. A circular silver rim sits on top of each bin. Even though the objects are viewed from different directions, their shape means that whatever the viewpoint, the objects always have a very similar appearance (as can be seen in a selection of examples shown in figure 4-64). The dataset is small with only 122 examples. Figure 4-65 shows that the range of extract dimensions is also quite limited and that all instances in the dataset are smaller than the fixed resizing dimensions.



Figure 4-64: Sample extracts from the Earthmine “Garbage Bin” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

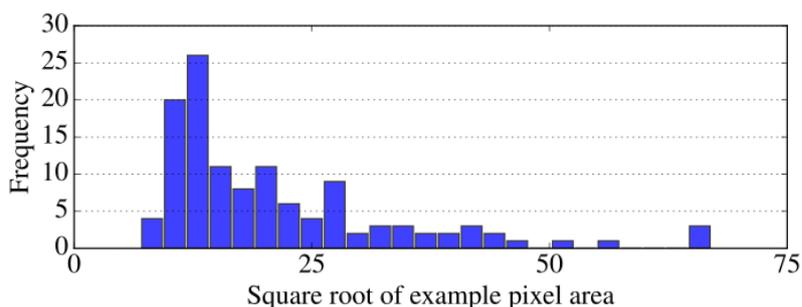


Figure 4-65: Distribution of example sizes for the Earthmine “Garbage Bin” object type.

Intensity Based Descriptors

The high degree of homogeneity in the examples means that accuracy is extremely good in absolute terms across all of the extractor configurations in both the linear and the non-linear classification cases. The precision-recall graphs shown in figure 4-66 show near perfect discrimination across the full range of classification thresholds between the object examples and the negative class of random extracts. Although all of the extractors do well, in the linear case shown in figure 4-66(a), the HOG descriptors give slightly lower recall accuracy over the Pro-HOG descriptors. The differences between the different Pro-HOG configurations are not

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9916	0.9672	0.9793
Pro-HOG (GSS, 72×72)	Linear	0.9959	0.9836	0.9897
Pro-HOG (72×72)	Linear	0.9959	0.9836	0.9897
Pro-HOG (GSS)	Linear	0.9918	0.9918	0.9918
Pro-HOG	Linear	0.9878	0.9918	0.9898
HOG	Non-Linear	1.0000	0.9672	0.9833
Pro-HOG (GSS, 72×72)	Non-Linear	1.0000	0.9877	0.9938
Pro-HOG (72×72)	Non-Linear	1.0000	0.9795	0.9896
Pro-HOG (GSS)	Non-Linear	0.9959	0.9836	0.9897
Pro-HOG	Non-Linear	0.9916	0.9631	0.9771

Table 4.22: Classification Results – Earthmine “Garbage Bin” (Intensity descriptors)

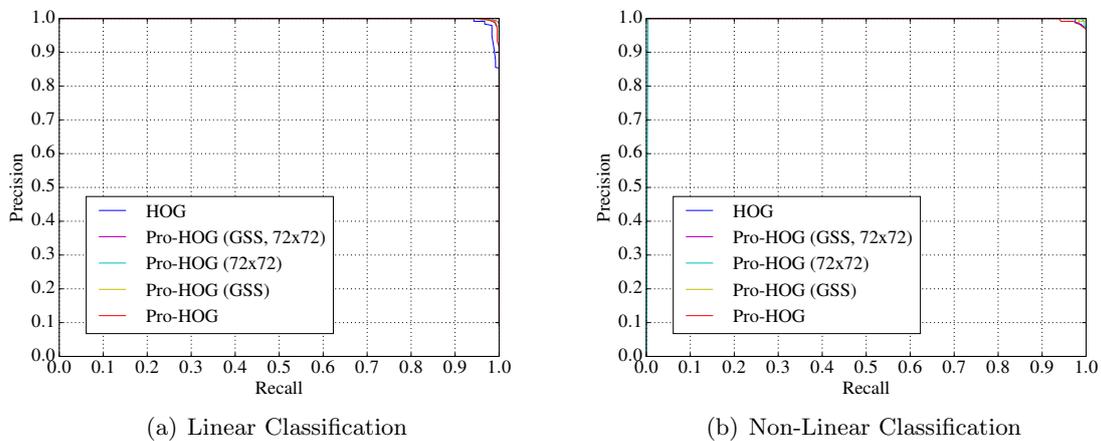


Figure 4-66: Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Garbage Bin” object type.

large enough to draw inferences about which of the configurations is best for this object type.

Depth Based Descriptors

As previously mentioned in the case of the Earthmine “Car” dataset, this object type also suffers from inadequate depth information. Figure 4-67 shows that in both the linear and the non-linear classification case, accuracy is much worse across all extractor types than when simply using the intensity based descriptors. Due to the small sizes of the objects, and the inaccuracies endemic to stereoscopy, some of the examples were even missing depth information entirely meaning that descriptors from those examples were unable to differentiate

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.3363	0.3074	0.3212
Pro-HOG (GSS, 72×72)	Linear	0.5503	0.4262	0.4804
Pro-HOG (72×72)	Linear	0.4400	0.3607	0.3964
Pro-HOG (GSS)	Linear	0.5698	0.4180	0.4823
Pro-HOG	Linear	0.4080	0.3361	0.3685
HOG	Non-Linear	0.8507	0.2336	0.3666
Pro-HOG (GSS, 72×72)	Non-Linear	0.8103	0.3852	0.5222
Pro-HOG (72×72)	Non-Linear	0.8974	0.2869	0.4348
Pro-HOG (GSS)	Non-Linear	0.8015	0.4303	0.5600
Pro-HOG	Non-Linear	0.7959	0.3197	0.4561

Table 4.23: Classification Results – Earthmine “Garbage Bin” (Depth descriptors)

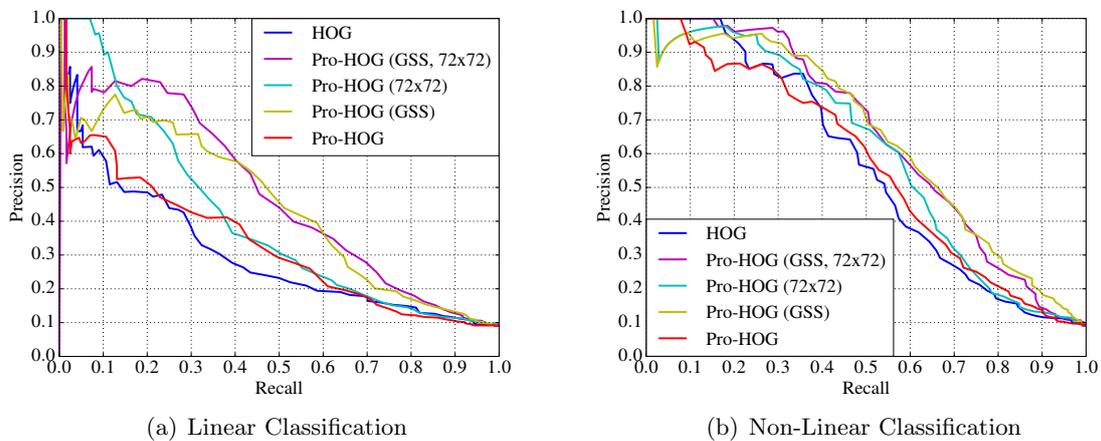


Figure 4-67: Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Garbage Bin” object type.

positive objects from negative extracts – especially from parts of the street scene where garbage bins were not present. This is the cause behind the low recall rates experienced.

In both the linear and the non-linear cases, while accuracy was low overall, there is large improvement in recall when using Pro-HOG with GSS enabled. Pro-HOG in its basic configuration is no better or worse than HOG overall, but it does trade slightly worse precision for better recall which indicates that Pro-HOG generalises slightly better for this object type using the depth data available.

4.3.16 Earthmine “Traffic Light”

Some examples of the Earthmine “Traffic Light” class are shown in figure 4-68. Figure 4-69 shows that more than 86% of the instances in this dataset are below the fixed resize dimensions of 72×72 pixels. The improved colourisation of the Earthmine dataset means that although the objects themselves are more distinct in their appearance than in the AAM “Traffic Light” class, the backgrounds are much more varied. Some examples also suffer from occlusions (*e.g.* people walking in front of the lights, or signs that are mounted on the light poles). Unlike the AAM “Traffic Light” class, nearly all of the examples show traffic lights atop a single central vertical pole. There are 178 examples in this dataset meaning that it is expected to be difficult given the significant intraclass variance to produce a well performing classifier whatever feature extractor is used.



Figure 4-68: Sample extracts from the Earthmine “Traffic Light” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

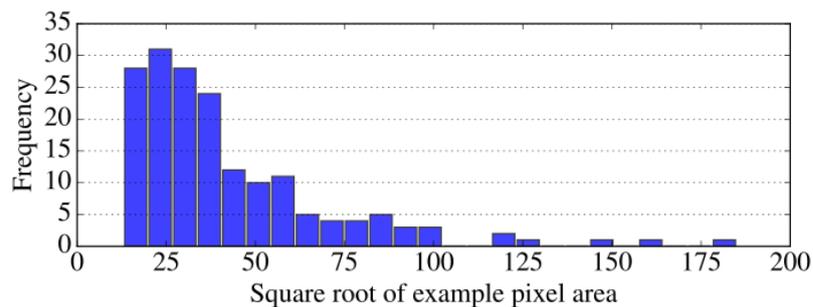


Figure 4-69: Distribution of example sizes for the Earthmine “Traffic Light” object type.

Intensity Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.7601	0.7011	0.7294
Pro-HOG (GSS, 72×72)	Linear	0.8158	0.7126	0.7607
Pro-HOG (72×72)	Linear	0.7276	0.6753	0.7004
Pro-HOG (GSS)	Linear	0.7778	0.7040	0.7391
Pro-HOG	Linear	0.6607	0.6322	0.6461
HOG	Non-Linear	0.9349	0.7011	0.8013
Pro-HOG (GSS, 72×72)	Non-Linear	0.8949	0.7586	0.8212
Pro-HOG (72×72)	Non-Linear	0.9111	0.7069	0.7961
Pro-HOG (GSS)	Non-Linear	0.9220	0.7471	0.8254
Pro-HOG	Non-Linear	0.8885	0.7098	0.7891

Table 4.24: Classification Results – Earthmine “Traffic Light” (Intensity descriptors)

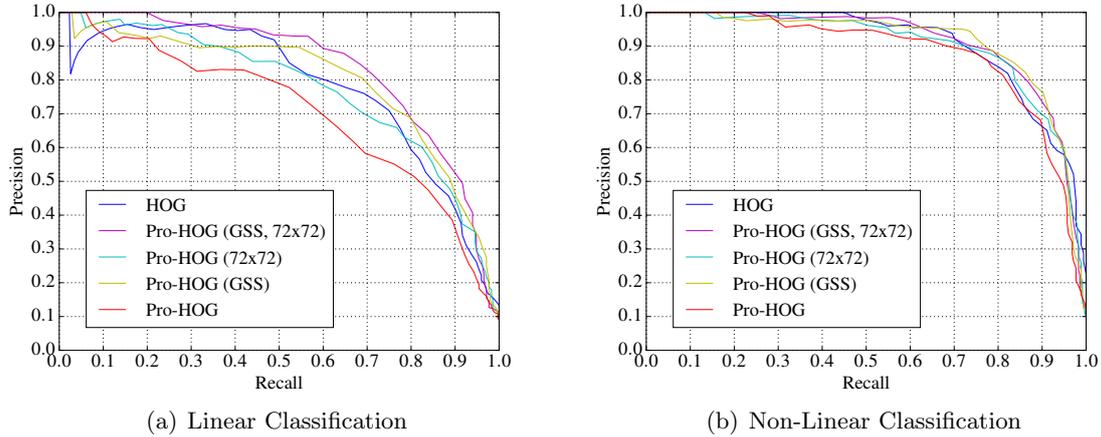


Figure 4-70: Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Traffic Light” object type.

In the linear classification case, figure 4-70(a) shows that there is a large amount of variation in the accuracy given by the different extractors. For both the resizing and the non-resizing configurations of Pro-HOG, having GSS enabled allows for much higher precision and recall. However, HOG, which does not implement GSS, has slightly better accuracy than the image resizing version of Pro-HOG which it is closest to in its configuration. HOG’s more involved process of spreading the contribution of each cell’s histogram values to its neighbouring cells appears to be better suited to classifying this object type.

In the non-linear case, figure 4-70(b) shows again that classification is higher overall across all five extractors and that the difference in accuracy between the different extractor types is much lower. The overall ordering between the different extractors is relatively unchanged in

the non-linear case.

For this object type using the intensity based descriptors, the results do not appear to show that Pro-HOG’s scale-independent feature extraction is helpful in improving classification accuracy.

Depth Based Descriptors

As in the previous Earthmine object categories, figure 4-71 shows that the inaccurate depth data results in very poor classification accuracy when using the depth based descriptors. With GSS enabled, Pro-HOG is much better than HOG in the linear case when resizing the images. The effect is less pronounced in the non-linear case and it is not clear why accuracy should improve so much only when these two algorithmic schemes of Pro-HOG are in effect together. On this object type, Pro-HOG appears to be more effective than HOG at deriving descriptors from degraded depth information.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.3292	0.3017	0.3148
Pro-HOG (GSS, 72×72)	Linear	0.5933	0.4569	0.5162
Pro-HOG (72×72)	Linear	0.5054	0.4023	0.4480
Pro-HOG (GSS)	Linear	0.5076	0.3851	0.4379
Pro-HOG	Linear	0.4502	0.3764	0.4100
HOG	Non-Linear	0.7586	0.2529	0.3793
Pro-HOG (GSS, 72×72)	Non-Linear	0.8282	0.3879	0.5284
Pro-HOG (72×72)	Non-Linear	0.8106	0.3075	0.4458
Pro-HOG (GSS)	Non-Linear	0.7576	0.3592	0.4873
Pro-HOG	Non-Linear	0.7188	0.2644	0.3866

Table 4.25: Classification Results – Earthmine “Traffic Light” (Depth descriptors)

4.3.17 Earthmine “Parking Sign”

The appearance variability in the Earthmine “Parking Sign” dataset is low. Each object is composed of a thin black pole surmounted by a small white rectangular sign. The signs are always placed at the kerbside and are rarely occluded given their proximity to the road itself. Some examples are shown in figure 4-72. The distance from these objects to the imaging equipment falls into bands according to whether the objects are placed at kerb edge nearest to the mapping vehicle, or at the kerb edge furthest from the vehicle (to the left or the right of the vehicle). This results in the two peaks in the size distribution graph shown in figure 4-73.

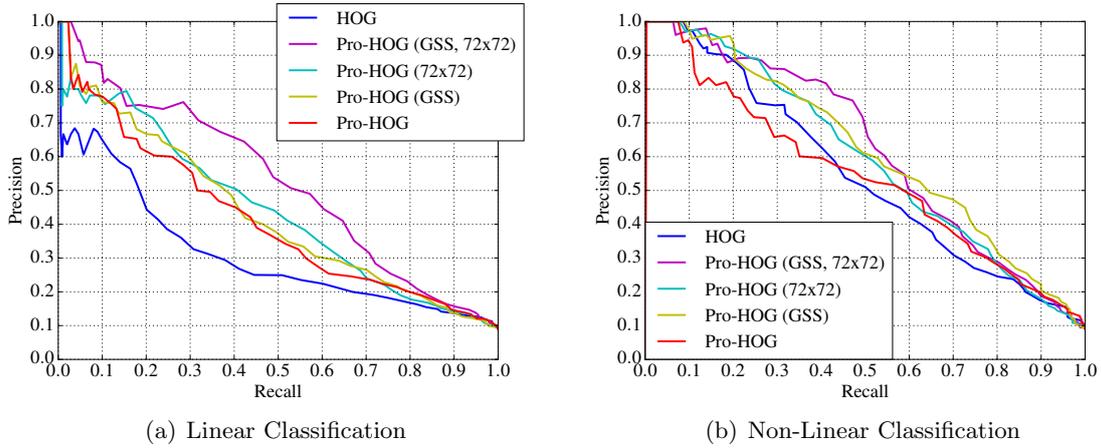


Figure 4-71: Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Traffic Light” object type.

For the examples closest to the imaging vehicle, the signs have more variance in apparent size resulting in the tail to the distribution. Nearly 96% of the objects are smaller than the fixed resize dimensions.

Intensity Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.9399	0.9301	0.9350
Pro-HOG (GSS, 72×72)	Linear	0.9468	0.9336	0.9401
Pro-HOG (72×72)	Linear	0.9399	0.9301	0.9350
Pro-HOG (GSS)	Linear	0.9565	0.9231	0.9395
Pro-HOG	Linear	0.9191	0.8741	0.8961
HOG	Non-Linear	0.9844	0.8846	0.9319
Pro-HOG (GSS, 72×72)	Non-Linear	0.9888	0.9301	0.9586
Pro-HOG (72×72)	Non-Linear	0.9889	0.9336	0.9604
Pro-HOG (GSS)	Non-Linear	0.9890	0.9406	0.9642
Pro-HOG	Non-Linear	0.9731	0.8846	0.9267

Table 4.26: Classification Results – Earthmine “Parking Sign” (Intensity descriptors)

The results for the different configurations of Pro-HOG shown in figure 4-74(a) have a similar relationship to those for the Earthmine “Traffic Light” class (although the accuracy is improved overall). Pro-HOG in its base configuration is less accurate overall by about four percentage points than the Pro-HOG variants with either GSS enabled, or when resizing the extracts. HOG is comparable in accuracy to Pro-HOG in these adjusted configurations, but has higher recall; Pro-HOG has higher precision.



Figure 4-72: Sample extracts from the Earthmine “Parking Sign” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

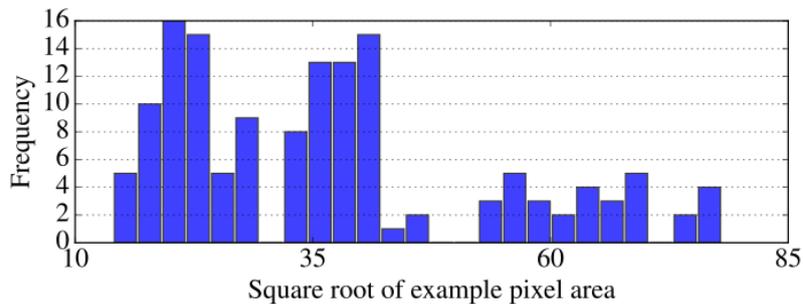


Figure 4-73: Distribution of example sizes for the Earthmine “Parking Sign” object type.

In the non-linear case, figure 4-74(b) shows that the extractors are much more similar in their levels of accuracy. Pro-HOG still gives slightly higher levels of recall in its three adjusted configurations over its base configuration however. As seen previously, accuracy overall in the non-linear case is much better than in the linear case.

Depth Based Descriptors

The Earthmine depth data for this object type are especially corrupt. In most examples, the only part of the object seen in the depth images is the sign itself; the vertical pole is not visible in the depth images of any of the extracts. In addition, the flat rectangular sign itself has

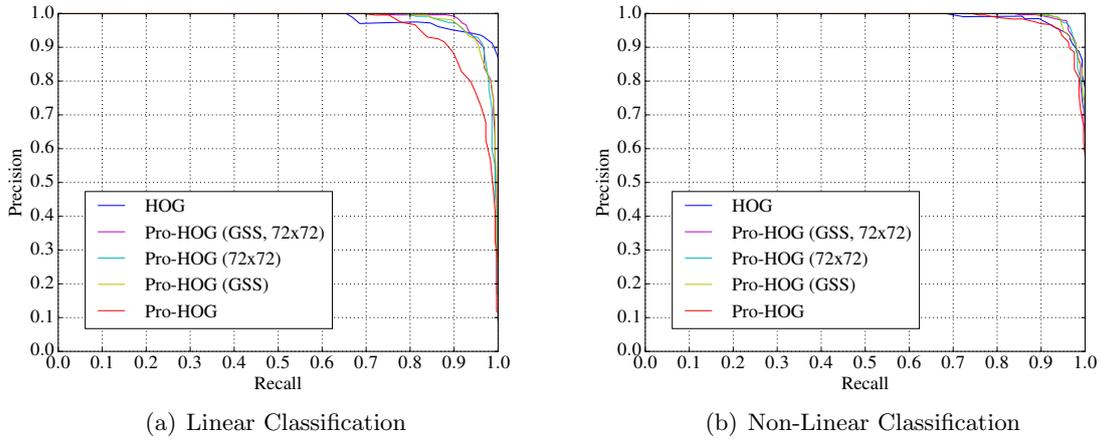


Figure 4-74: Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Parking Sign” object type.

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.3100	0.2483	0.2757
Pro-HOG (GSS, 72×72)	Linear	0.4481	0.2867	0.3497
Pro-HOG (72×72)	Linear	0.4360	0.3217	0.3702
Pro-HOG (GSS)	Linear	0.4806	0.3462	0.4024
Pro-HOG	Linear	0.3858	0.3601	0.3725
HOG	Non-Linear	0.5673	0.2063	0.3026
Pro-HOG (GSS, 72×72)	Non-Linear	0.6074	0.3462	0.4410
Pro-HOG (72×72)	Non-Linear	0.5972	0.3007	0.4000
Pro-HOG (GSS)	Non-Linear	0.6641	0.3042	0.4173
Pro-HOG	Non-Linear	0.6923	0.2832	0.4020

Table 4.27: Classification Results – Earthmine “Parking Sign” (Depth descriptors)

no interesting surface morphology and the rectangular silhouette of the sign is often poorly defined in the data. All of these issues mean that the depth descriptors give very poor accuracy when combined with the intensity based descriptors across all of the different extractor types, in either the linear, or the non-linear classification tasks (as seen in figure 4-75).

Between the differing configurations of Pro-HOG, there are no great differences in classification accuracy although the GSS enabled versions show a slight improvement over the GSS disabled versions. Even though the depth data for this object type are poor, it is clear in both the linear and the non-linear results that the Pro-HOG extractors create descriptors that allow for greater precision and recall to be attained over the HOG extractor. This is likely due to the more complex interpolation of adjacent cell histograms in HOG. In Pro-HOG, because the values of adjacent cell histograms are not interpolated, there is less chance of missing in-

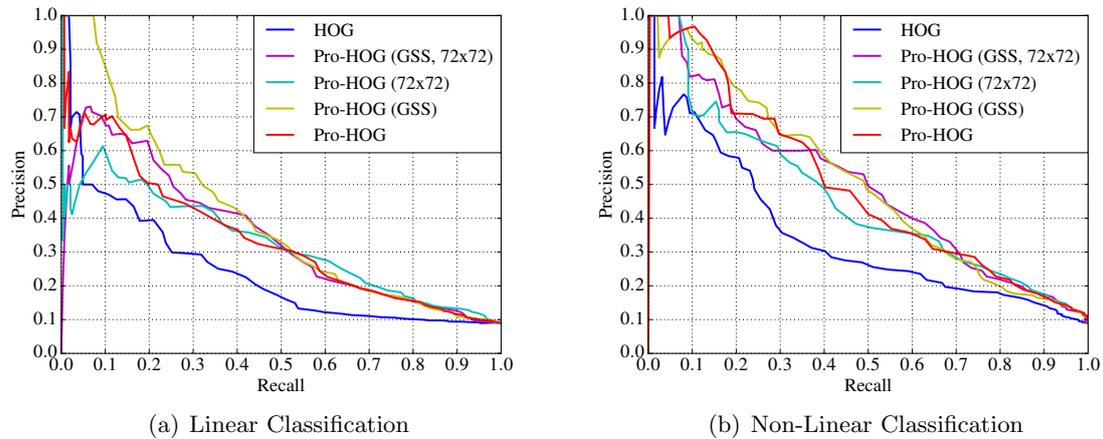


Figure 4-75: Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Parking Sign” object type.

formation in one cell corrupting valid information in another. In HOG, the assumption that useful features are present in all parts of the bounding rectangle specifying the extract means that spurious or missing features can negatively impact on the ability of the final descriptor to accurately represent the object. This makes Pro-HOG more robust, but it also potentially means that it suffers in its generalisability since more emphasis is given to the presence (or absence) of features localised to a single cell.

4.3.18 Earthmine “Traffic Cone”

The Earthmine “Traffic Cone” class is another vertically symmetric object type. The most salient feature of this object — its distinctive orange colour — is not passed to the feature extractors since images are first converted to grey scale. This object type’s size, simplicity, and lack of distinguishing characteristics (in grey scale) suggest that it has the potential to be easily confused for background scene elements. However, the ground-truth samples for this type have minimal intraclass variance meaning that the classifier is also less likely to be confused. Some examples of the type are shown in figure 4-76. Figure 4-77 shows that the small size of the objects means that the dimensions of the image extracts are all less than HOG’s fixed resizing dimensions. The objects are also indistinct especially where they appear in front of background having strong contrast edges (*e.g.* road kerbs and road markings) which increases the possibility of false-positive classifications.



Figure 4-76: Sample extracts from the Earthmine “Traffic Cone” dataset. Images are resized to fixed width for display here, but have varying actual dimensions.

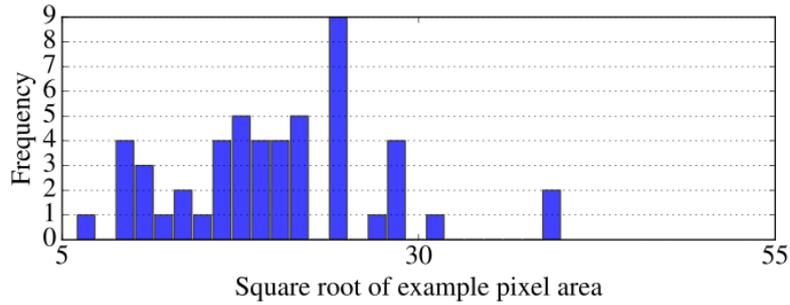


Figure 4-77: Distribution of example sizes for the Earthmine “Traffic Cone” object type.

Intensity Based Descriptors

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.8958	0.8431	0.8687
Pro-HOG (GSS, 72×72)	Linear	0.9175	0.8725	0.8945
Pro-HOG (72×72)	Linear	0.9032	0.8235	0.8615
Pro-HOG (GSS)	Linear	0.8571	0.7647	0.8083
Pro-HOG	Linear	0.8736	0.7451	0.8042
HOG	Non-Linear	0.9535	0.8039	0.8723
Pro-HOG (GSS, 72×72)	Non-Linear	0.9535	0.8039	0.8723
Pro-HOG (72×72)	Non-Linear	0.9545	0.8235	0.8842
Pro-HOG (GSS)	Non-Linear	0.9318	0.8039	0.8632
Pro-HOG	Non-Linear	0.9383	0.7451	0.8306

Table 4.28: Classification Results – Earthmine “Traffic Cone” (Intensity descriptors)

In the linear classification case, figure 4-78(a) shows that both of the image resizing versions of Pro-HOG are similar to HOG in their precision and recall scores. GSS enabled Pro-HOG has the highest accuracy overall. The scale-independent versions of Pro-HOG (both the GSS enabled and disabled versions) do very badly on this object type. This is likely because the 8×8 cell-block dimensions being used are too large for the native pixel dimensions of the extracts. That is, the number of pixels being used by each cell to construct its histogram is too small to generate histograms that are stable enough in their representations across images. Additionally, because the cell pixel dimensions are not equal because the cell grid dimensions

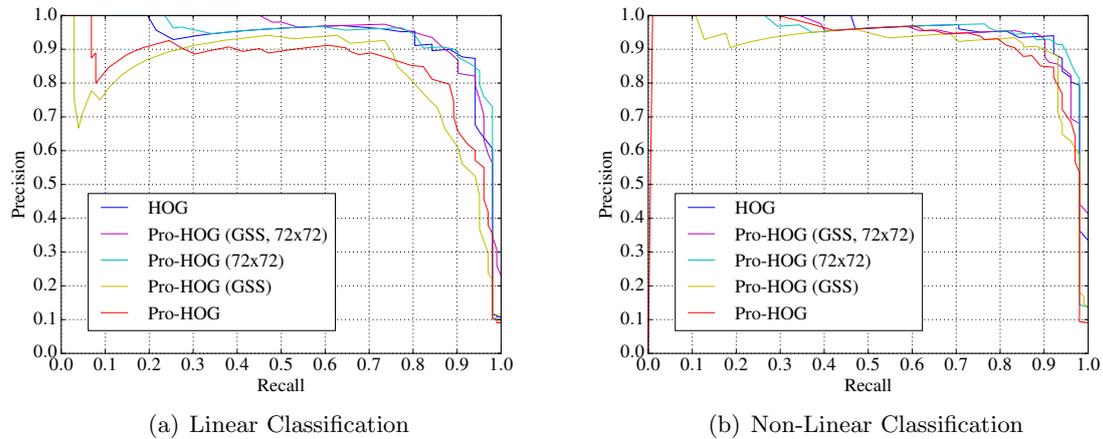


Figure 4-78: Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Traffic Cone” object type.

rarely divide the image dimensions by a whole number, in very small images there is a wider variation in the number of pixels that each cell references to construct its histogram. The histograms are then normalised and in doing so this can unfairly weight certain subsections of the extracts. Extracting descriptors over the entire object dataset, leads to inconsistencies between the cell histograms of spatially corresponding image/object parts.

In the non-linear case (see figure 4-78(a)), this issue still exists, but the flexibility in how the classification boundary is defined allows the classifier to be less influenced by these histogram encoding issues. The problem is exacerbated by the small size of the training dataset – the poor quality examples where the histogram generation issue is most problematic have a greater influence in deciding how the classification boundary should be placed. This is an example of overfitting to the training data.

To verify that this issue really is the cause of the poor accuracy seen in the non-resizing versions of Pro-HOG, the experiment was carried out again using cell-block dimensions of 4×4 (instead of 8×8). For HOG, each cell’s pixel dimensions are left at 8×8 pixels. For the resizing versions of Pro-HOG, this is also the pixel dimensions of the cells in those extractors. This implies extract resizing dimensions of 40×40 pixels. Using smaller cell block dimensions gives descriptors with only a quarter of the length – 576 elements instead of 2304. The precision-recall graphs for the linear and the non-linear classification cases using this configuration are shown in figure 4-79.

Comparing figures 4-78(a) and 4-79(a), using smaller cell grid dimensions for this particular object type results in improved recall and precision for the scale invariant configurations of

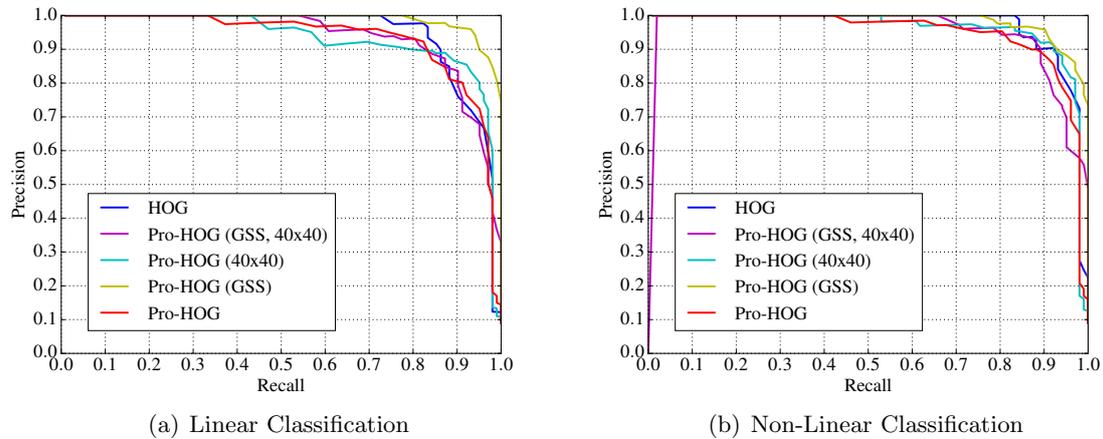


Figure 4-79: Precision versus Recall using feature descriptors extracted from intensity images of the Earthmine “Traffic Cone” object type with extraction cell-block dimensions of 4×4 .

Pro-HOG while the image resizing configurations (along with HOG) have relatively unchanged levels of accuracy.

This relative improvement in accuracy is also noticeable in the non-linear classification case as seen in figures 4-78(b) and 4-79(b) although the extent of improvement is less. By choosing a configuration of Pro-HOG that is better suited to the object type, greater classification accuracy is achieved in the linear case, than in the non-linear case. This is preferential for object detection because linear classification is much more efficiently computed. This result also demonstrates that larger feature vectors don’t necessarily result in feature encodings that give higher classification accuracy.

Depth Based Descriptors

For comparison against the initial experiment with intensity based descriptors (with results shown by the precision-recall curves in figure 4-78), the depth descriptors use the same 8×8 cell block dimensions. The quality of the depth information for this object type is again very degraded; the smaller size of the objects and the lower precision of stereoscopy in being able to capture detailed changes in depth means that very few of the objects have depth information that is representative of their morphology.

In the linear case, there is great variability in the accuracy achieved by the different extractor configurations (see figure 4-80(a)). The two best performing extractors are Pro-HOG with GSS enabled – both with and without resizing the extracts. The worst performing extractor is

Feature Extractor	Classifier	Precision	Recall	F1
HOG	Linear	0.6479	0.4510	0.5318
Pro-HOG (GSS, 72×72)	Linear	0.7143	0.5882	0.6452
Pro-HOG (72×72)	Linear	0.6164	0.4412	0.5143
Pro-HOG (GSS)	Linear	0.6889	0.6078	0.6458
Pro-HOG	Linear	0.5200	0.3824	0.4407
HOG	Non-Linear	1.0000	0.3725	0.5429
Pro-HOG (GSS, 72×72)	Non-Linear	1.0000	0.5882	0.7407
Pro-HOG (72×72)	Non-Linear	0.9744	0.3725	0.5390
Pro-HOG (GSS)	Non-Linear	0.9265	0.6176	0.7412
Pro-HOG	Non-Linear	0.9286	0.3824	0.5417

Table 4.29: Classification Results – Earthmine “Traffic Cone” (Depth descriptors)

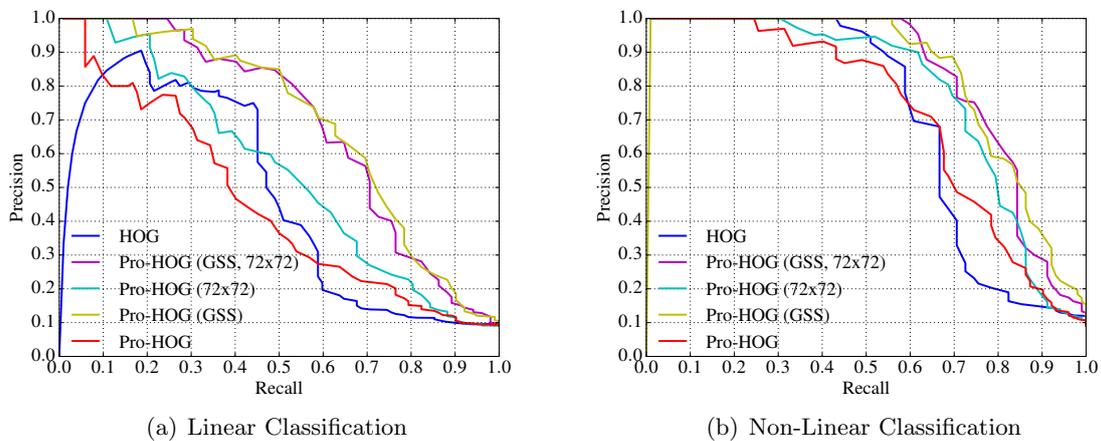


Figure 4-80: Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Traffic Cone” object type.

Pro-HOG in its basic configuration (*i.e.* with GSS disabled). The configuration of Pro-HOG that is most similar to HOG is also closest in accuracy to HOG.

In the non-linear case, figure 4-80(b) shows that all of the extractors exhibit increased accuracy. The relative ordering of the Pro-HOG extractors is unchanged; the GSS enabled versions still give the highest accuracy. In the non-linear case, HOG’s accuracy is relatively lower compared with the two GSS disabled configurations of Pro-HOG implying that Pro-HOG gives the learning algorithm more flexibility in being able to find a suitable classification boundary.

In light of the increase in accuracy using the Pro-HOG extractors with the intensity based descriptors when reducing the cell block dimensions to 4×4 (with results seen in figure 4-79),

the experiment was repeated using the depth based descriptors and 4×4 cell blocks. The results of these experiments are shown by the precision-recall curves for the linear and the non-linear cases in figure 4-81.

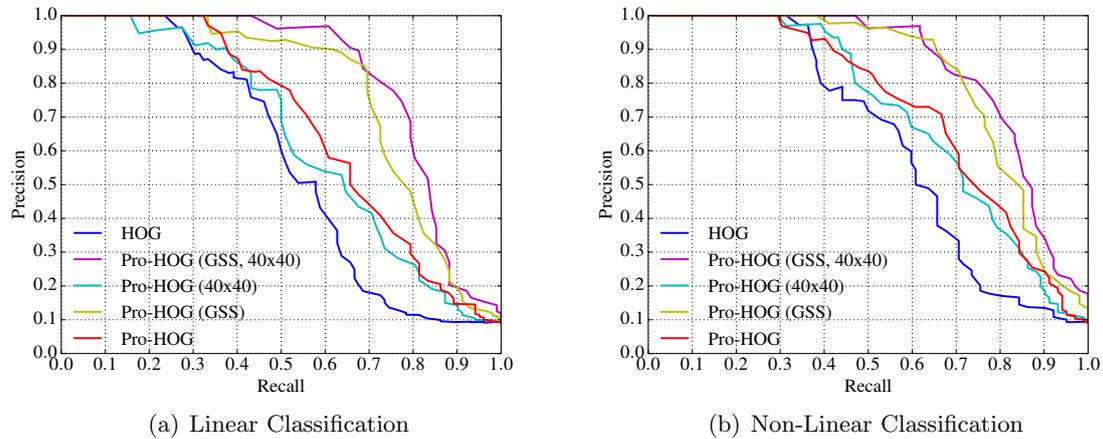


Figure 4-81: Precision versus Recall using feature descriptors extracted from depth images of the Earthmine “Traffic Cone” object type with extraction cell-block dimensions of 4×4 .

When comparing figure 4-80(a) with figure 4-81(a), and figure 4-80(b) with figure 4-81(b), it is clear that accuracy is also improved using the depth descriptors when going from cell block dimensions of 8×8 to 4×4 . While HOG as well as Pro-HOG in all its configurations gives greater accuracy, the improvement is relatively larger for Pro-HOG. In both the linear and the non-linear tests, all of the Pro-HOG configurations now provide conclusively better accuracy over HOG.

Also notable when comparing figures 4-81(a), and 4-81(b) is the fact that non-linear classification, while still giving greater accuracy than linear classification, does not allow for as great an improvement as seen in the initial depth based descriptor experiment (with results shown in figure 4-80) where cell block dimensions of 8×8 are used. The reason for this is that a smaller number of feature vector dimensions constrains the flexibility of the classification boundary that can be learned for linear classification (the gain in accuracy in the linear case is much greater than in the non-linear case); for the same size training dataset, the learner is less likely to fit a classification hyperplane to the data that gives poor generalisation on query data. This is an example of how the *curse of dimensionality* can degrade classification accuracy.

4.4 Observations and Recommendations

Overall, the results of the single object binary classification experiments show that there is no standard configuration of any of the extractors (HOG or Pro-HOG) that is well suited for all of the different object types. The size and symmetry of the objects are two factors which influence the quality of the extracted feature vectors. Ideally, for each object classification task, the feature extractors being evaluated should be specifically configured using cross validation over a subset of the object’s dataset. This is a computationally intensive process that takes exponentially longer with the addition of subsequent extractor configuration parameters. As demonstrated on the Earthmine “Traffic Cone” class, optimising classification accuracy for one type of object cannot be achieved by choosing parameters for the extractors that are based on the cross-validation results of object types that are significantly dissimilar.

4.4.1 Emulating HOG

In the preceding tests, HOG is included as a baseline feature extractor to confirm that Pro-HOG gives similar classification results. The second configuration of Pro-HOG tested in all of the experiments emulates HOG as closely as possible. This is achieved by resizing the image examples to the same fixed dimensions that HOG uses, and disabling GSS (to reflect the fixing of this parameter in the OpenCV implementation being used). When emulating HOG in this way, the differences between the two algorithms are Pro-HOG’s lack of cell grid resolution Gaussian filtering, and Pro-HOG’s lack of bilinear spatial interpolation of the cell histograms which cannot be incorporated into Pro-HOG due to the division of the algorithm into two separate phases.

Given that Pro-HOG does not implement these aspects of the HOG algorithm, Pro-HOG in its HOG emulation configuration is not expected to meet or exceed the classification accuracy afforded by HOG. However, in twelve out of the fifteen linear classification experiments, Pro-HOG exceeded HOG in classification accuracy (measured as the F1 score) by an average of 1.33% (with standard deviation of 2.57%). In the non-linear classification experiments, Pro-HOG’s overall classification accuracy also exceeded HOG’s, but the improvement was smaller with a mean gain in the F1 score of 0.81% (standard deviation of 2.51%).

This difference has two explanations. Either the OpenCV implementation of HOG deviates from the specifications laid out in Dalal and Triggs (2005) in undocumented ways, or HOG’s cell resolution Gaussian filtering and/or bilinear histogram spatial interpolation is acting to decrease classification accuracy rather than increase it for most of the object types. Since HOG

was originally evaluated against only a single object type – the INRIA person dataset (Dalal and Triggs, 2005) – it is possible that certain features of the algorithm do not generalise to different object types as well as originally intended. Follow-up investigation can help to understand which of these explanations is more likely.

4.4.2 Effects of Gradient Sign Sensitivity (GSS)

It was not expected that GSS would be so effective in improving classification accuracy, particularly because all of the experiments were carried out using the left-right reflections of the extracts. In 14 of the 15 object types, enabling GSS resulted in higher overall classification accuracy although often improving precision at the expense of recall. The only object type where enabling GSS resulted in lower classification accuracy was the AAM “Rectangular Road Sign” case, and there is no clear reason why enabling GSS should be less effective for this object type especially when classification accuracy is improved with GSS enabled for similar object types such as the Earthmine “Parking Sign” and AAM “Triangular Road Sign” object types. This only further supports the notion that the extractor must be configured for the given object type; presuppositions about the extractor’s “best” parameters for a given object type based on evaluations of similar object types will not always hold.

4.4.3 Effects of Scale Invariance (SI)

These experiments demonstrate the effect of scale invariant feature extraction at fixed cell-block dimensions. By comparing the classification accuracy of Pro-HOG in its originally designed scale invariant mode, versus its HOG emulating resizing mode, all other aspects of the algorithm are controlled to remain equivalent and the effect of image resizing on classification accuracy can be seen.

Figure 4-82 shows the percentage improvement in classification accuracy (measured as the F1 score) of Pro-HOG when extracting intensity based features in scale independent mode at the fixed cell-block dimensions of 8×8 over all of the object types. The green bars show improvement with GSS enabled and the blue bars show improvement with GSS disabled. The object types are ordered in the X axis in increasing median image dimensions.

Figure 4-82 firstly indicates that Pro-HOG’s scale independent feature extraction does not in general improve upon the classification accuracy achieved in Pro-HOG’s image resizing HOG emulation mode at the given cell-block and histogram length configuration. Secondly, im-

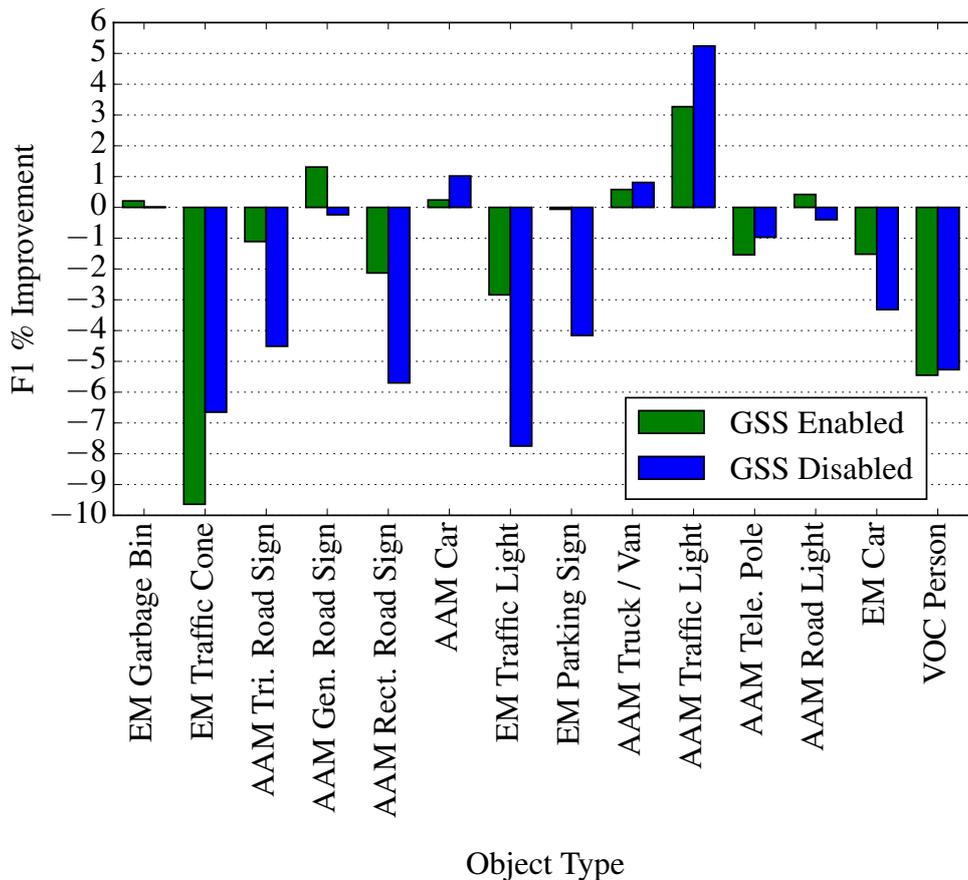


Figure 4-82: Percentage improvement in accuracy (F1 score) of Pro-HOG (Scale Independent) from Pro-HOG (Resizing) over all object classes at cell-block dimensions of 8×8 .

Improvements in classification accuracy due to Pro-HOG’s scale independent feature extraction are not correlated (in either the GSS enabled or disabled case) with the original size of the extracts.

These experiments fixed the cell-block dimensions to 8×8 so as to ascertain how the scale-invariant mode of Pro-HOG changes the quality of extracted features and so affects classification accuracy. The dimensions were chosen because they were found in section 4.2.5 to allow for good classification accuracy on the Pascal VOC “Car” and “Person” object types. In addition, the same dimensions are used by Dalal and Triggs (2005) in the evaluation of HOG on the INRIA person dataset. However, as was seen in the Earthmine “Traffic Cone” experiment in section 4.3.18, decreasing the cell-block dimensions for SI Pro-HOG improves classification accuracy to be better than any of the other extractors using the 8×8 cell-block dimensions (with the intensity based descriptors). The reason for this is that the size of the images in that dataset are already very small. At lower cell-block dimensions, the number of pixels per

cell increases and each cell can therefore contribute more information to its histogram; the constructed histograms are less “coarse” than when using larger cell-block dimensions and so the descriptors correspond more closely to one another.

4.4.4 Better Accuracy with Shorter Descriptors

For the majority of the objects tested, the average size of the images is less than the fixed resize dimensions of 72×72 pixels. At cell-block dimensions of 8×8 (giving a cell grid resolution of 9×9), this means that Pro-HOG in its original (non-resizing) configuration, constructs cell histograms using fewer than 8^2 pixels per cell. This means that it should be the case for most of the object types that better classification accuracy can be achieved if lower cell-block dimensions are used – even though this reduces the overall size of the feature vectors being generated.

To test this hypothesis, a new set of tests to determine the effect of scaling on accuracy was undertaken for each of the object types. Only linear classification is used since, as the preceding results have shown, the differences in extractor behaviour are more obvious in the linear classification case than in the non-linear case. Only intensity based descriptors are used for these experiments since there is no reason to think that the use of depth descriptors would affect the hypothesis. In addition, the tests on the Earthmine “Traffic Cone” object type in section 4.3.18 showed that accuracy is also improved for that object type using the depth based descriptors when reducing the cell grid resolution. GSS is also left disabled since it is found to give lower overall accuracy for most of the object types – giving more room for improvement. This also means that Pro-HOG can be directly compared to HOG which fixes GSS to be disabled in its tested implementation. The histogram length is left unchanged at nine.

For each object type, Pro-HOG is evaluated three times using five fold cross validation. Initially, Pro-HOG in SI mode is used to determine the cell-block dimensions for the object type that give the best classification accuracy (evaluated as the F1 score). In the subsequent experiments, Pro-HOG is configured to use these cell-block dimensions, but the images are resized prior to feature extraction.

In the first experiment, the fixed width and height of the extracts are set as the square root of the median area of the examples in the object dataset. This means that as many images are scaled up as are scaled down. In the second experiment, the fixed width and height are set to be the minimum size that is at least as large in either dimension as the largest example

in the dataset, but is also a whole multiple of the minimum cell pixel dimensions that allow this to be true. For example, for some object datasets, if h is found to be the image with the largest dimension (either horizontally or vertically), and if b is found to be the cell-block dimensions that give optimal classification accuracy in SI Pro-HOG, then the image resize dimensions $m_b \times m_b$ are found as

$$n_b = \left\lceil \frac{\max(h_x, h_y)}{b + 1} \right\rceil \quad (4.7)$$

$$m_b = (b + 1)n_b \quad (4.8)$$

with $\max(x, y)$ defined as the larger of its two parameters. $b + 1$ is used because cell-block dimensions of $b \times b$ require a cell grid with resolution $b + 1$ on a side. For comparison, HOG is also evaluated using the optimal cell-block dimensions, but with predetermined cell pixel dimensions of $n_b \times n_b$ which ensures that HOG scales the images prior to extraction to $m_b \times m_b$ pixels – the same dimensions as for Pro-HOG in its maximum resizing configuration.

The two different resizing settings are used to see how feature extraction quality changes. In the first experiment, for each object type the same number of images are scaled up as are scaled down. The median size for each object type is used as the fixed resizing dimensions because scaling affects the extraction process differently depending on whether the scaling factor is positive and < 1 or > 1 . Using the median size removes the possibility of bias caused by some object types having a larger average size than others.

In the second experiment, all of the images are scaled up. This is the best case for an extractor that undertakes image resizing because any scaling down causes information to be lost as shown by Dollar *et al.* (2014), potentially reducing the ability of the classifier to use the features as they are encoded to discriminate between the positive and negative examples. Scaling up does not lose information, but scaling artefacts may still be introduced causing some degradation in the quality of the features that are extracted. It is expected that scaling up all of the images should allow for improved classification accuracy than when scaling up half of the images and scaling down the other. If scaling introduces any kind of image corruption, then SI Pro-HOG should give classification accuracy that is at least as high as in the maximum image resizing case.

Tables 4.30, 4.31, and 4.32 show the results of these experiments grouped by parent dataset (Pascal VOC 2007, AAM and Earthmine). The first column gives the object type with the size of the feature descriptor given in parentheses. The size of the descriptor depends upon the optimal setting of the cell-block dimensions found for the object type by the SI variant of Pro-HOG whose results are shown in the first row for the object. Column FX (Feature

Extractor) specifies the configuration of the experiment where “Median” indicates that the images are first resized to the median example size, and “Max” which indicates that the images are first resized to the maximum dimensions for the object type. The dimensions that the images are resized to prior to feature extraction are shown in parentheses (the original “native” dimensions of the extracts apply in the SI case since, by definition, no image scaling is performed). The classification accuracy values are given at the default classification threshold. Precision-recall graphs are not shown due to the similarity of results between the different extractors.

Object (FV Dims.)	FX (Image Dims.)	Precision	Recall	F1
Car (1764)	SI (native)	0.8806	0.8196	0.8490
	Median (122 × 122)	0.8802	0.7915	0.8335
	Max (472 × 472)	0.8937	0.8018	0.8453
	HOG (472 × 472)	0.9080	0.8059	0.8539
Person (3600)	SI (native)	0.7557	0.4083	0.5301
	Median (104 × 104)	0.7472	0.4465	0.5590
	Max (440 × 440)	0.7606	0.4443	0.5609
	HOG (440 × 440)	0.7032	0.4733	0.5658

Table 4.30: Pascal VOC 2007 dataset: comparing Pro-HOG’s scale independence versus image resizing using optimal cell-block dimensions.

For the Pascal VOC 2007 “Car” class, the percentage improvements in precision for the SI and maximum resizing versions of Pro-HOG are 0.05% and 1.53% respectively. The improvements in recall for the same class are 3.55% and 1.30% respectively. The overall improvement as measured by the F1 score for SI Pro-HOG is 1.86%. For the maximum resizing version of Pro-HOG the improvement is 1.42%. The optimal setting for the cell-block dimensions in this case (7 × 7) results in a smaller feature vector but allows for higher accuracy than in the original experiment in section 4.3.4.

For the Pascal VOC 2007 “Person” class, the percentage improvements in precision for the SI and maximum resizing versions of Pro-HOG are 1.14% and 1.79% respectively. Recall is *reduced* with SI Pro-HOG by 8.56%. Max Pro-HOG also reduces recall accuracy, but only by 0.49%. Overall with this class, SI Pro-HOG reduces overall accuracy by 5.17%. Max Pro-HOG increases accuracy slightly by 0.34%. Contrary to expectations, in this case the resizing of all of the images to median dimensions actually causes accuracy to increase overall; both the maximum resizing and SI variants of Pro-HOG do worse in recall – although SI Pro-HOG does much worse. The Pascal VOC 2007 “Person” class is the largest of all of the datasets and so the results have significant statistical weight. The most likely explanation for this discrepancy is that in order to achieve the higher overall classification accuracy, the optimal cell-block dimensions of 10 × 10 are actually larger than used in the original experiment in section 4.3.5. Because a large number of the examples in this class are significantly smaller

than the median image dimensions, this means that the number of pixels per cell being used to construct each histogram is smaller. This object type also has a particularly high degree of intraclass variance as evidenced by the very low classification accuracy. This may be an example of an object class where scaling the images up improves classification accuracy because very finely detailed texture (as might be present in clothing) is not as accurately encoded without being sampled over a larger number of pixels.

As shown in table 4.30, HOG has the highest classification accuracy for both of the classes tested in the Pascal VOC 2007 dataset, although recall is highest for the “Car” object type using SI Pro-HOG. For the “Person” class, HOG achieves much higher recall, but much lower precision. The improved recall for the “Person” class may be caused by the extra functionality present in that algorithm which serves to smooth small differences in appearance – a feature that is better suited to this object type in particular.

Table 4.31 and figures 4-83 and 4-84 show the corresponding results for the object types from the AAM dataset. Figure 4-83 shows that precision is definitively improved in general with the one exception being the AAM “Traffic Light” case where precision is slightly degraded from the Median resizing case. SI Pro-HOG does not improve upon precision as much as the maximum resizing case however. The generally smaller sizes of the images in this class mean that SI Pro-HOG is in some instances building histograms of *negative* instances from fewer pixels per cell making the histograms coarser (having less variability to them) so that they are more easily confused as positive examples by the classifier (remembering that the negative instances have similar pixel dimensions to the positive instances in the dataset). This explanation still allows for the grouping together of positive examples in feature space as evidenced by the generally higher recall accuracy for SI Pro-HOG in figure 4-84.

In general, recall is improved in both the SI Pro-HOG and Max resizing cases over the Median resizing case (with very slight degradation in only two of the cases). In the case of the “Road Light” class, the cell-block dimensions are higher (at 9×9) than originally tested (see section 4.3.10), and so the reason for the lower recall values is almost certainly the same reason that explains why the classification accuracy for the Pascal VOC 2007 “Person” is degraded. Unexpectedly, recall is also degraded in the AAM “Traffic Light” case when resizing the images to maximum dimensions implying that losing some information in the positive images is beneficial for recall. It may be the case that the images from this dataset express a kind of appearance variability that is easily constrained by scaling the images down.

Overall, apart from the AAM “Road Light” object type, the optimal cell-block dimensions for SI Pro-HOG are lower than the cell-block dimensions used in the corresponding evaluations through section 4.3 leading to feature descriptors having a significantly smaller number of

dimensions. Table 4.31 shows that even though the feature descriptors are smaller, for every object type in the AAM dataset, SI Pro-HOG outperforms HOG and both resizing versions of Pro-HOG in recall.

Object (FV Dims)	FX (Image Dims.)	Precision	Recall	F1
Car (900)	SI (native)	0.9267	0.8841	0.9049
	Median (33×33)	0.9162	0.8471	0.8803
	Max (162×162)	0.9334	0.8560	0.8930
	HOG (162×162)	0.9546	0.8512	0.8999
Traffic Light (1296)	SI (native)	0.8828	0.6971	0.7790
	Median (46×46)	0.8831	0.6540	0.7515
	Max (245×245)	0.9025	0.6438	0.7515
	HOG (245×245)	0.9021	0.6293	0.7414
Triangular Road Sign (576)	SI (native)	0.8973	0.8037	0.8479
	Median (23×23)	0.8905	0.7485	0.8133
	Max (205×205)	0.9233	0.8129	0.8646
	HOG (205×205)	0.9137	0.7791	0.8411
Truck / Van (900)	SI (native)	0.8771	0.7021	0.7799
	Median (45×45)	0.8571	0.6223	0.7211
	Max (216×216)	0.8828	0.6410	0.7427
	HOG (216×216)	0.8415	0.5505	0.6656
Road Light (2916)	SI (native)	0.9477	0.8990	0.9227
	Median (89×89)	0.9472	0.9021	0.9241
	Max (350×350)	0.9572	0.9021	0.9288
	HOG (350×350)	0.9549	0.8977	0.9254
Telegraph Pole (144)	SI (native)	0.9433	0.9413	0.9423
	Median (48×48)	0.9237	0.9109	0.9172
	Max (273×273)	0.9262	0.9312	0.9287
	HOG (273×273)	0.9185	0.8965	0.9074
Rectangular Road Sign (324)	SI (native)	0.9142	0.8219	0.8656
	Median (30×30)	0.9036	0.7717	0.8325
	Max (216×216)	0.9295	0.7994	0.8596
	HOG (216×216)	0.9072	0.7171	0.8010
Generic Road Sign (576)	SI (native)	0.9198	0.7928	0.8516
	Median (27×27)	0.9134	0.7512	0.8244
	Max (195×195)	0.9377	0.7942	0.8600
	HOG (195×195)	0.9183	0.7287	0.8126

Table 4.31: AAM dataset: comparing Pro-HOG’s scale independence versus image resizing using optimal cell-block dimensions.

Table 4.32 and figures 4-85 and 4-86 show the results for the objects from the Earthmine datasets. As already noted, the number of instances per object type is very much lower than for the object types from the Pascal VOC 2007 and AAM datasets. As such, the results from these experiments carry less weight as they are more affected by the particular sampling of the data in each object class.

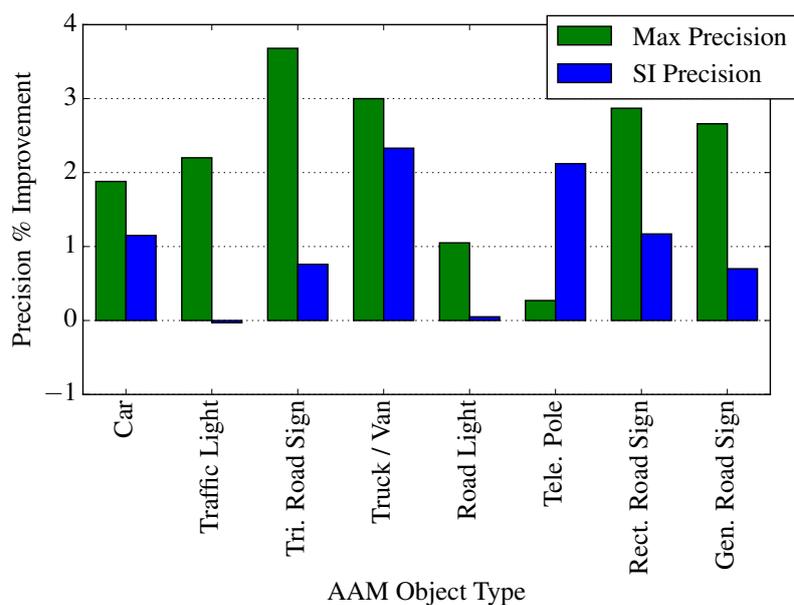


Figure 4-83: Percentage improvement in precision of the Pro-HOG (Scale Independent) and Pro-HOG (Max Resized) extractors from the Pro-HOG (Median Resized) extractor for object types from the AAM dataset (at object specific optimal cell-block dimensions).

Figure 4-85 shows that precision is again improved in general, but the smaller number of samples means that the results are less convincing than in the case of the AAM results. In two of the five object categories (“Garbage Bin” and “Parking Sign”), precision is degraded but not greatly. For the “Garbage Bin” object type, the optimal cell-block dimensions are 8×8 ; the same dimensions used in the original experiment in section 4.3.15.

In figure 4-86, recall is not significantly affected from the median resizing case overall, although there is variation across the object types – again, best explained by the reduced sample sizes. In the Earthmine “Car” case, a significantly larger cell-block dimensionality of 15×15 was needed to maximise classification accuracy for SI Pro-HOG, even though this necessarily reduced the number of pixels available to construct each cell’s histogram to less than 6^2 on average for half of the examples. If the hypothesis of this section is correct, this is a possible explanation for the drop in recall accuracy.

In the original Earthmine “Traffic Cone” experiment in section 4.3.18, while classification accuracy was greatly improved in the GSS enabled scale invariant Pro-HOG case at cell-block dimensions of 4×4 , the experiment here disables GSS. With this change, the optimal cell-block dimensions were found to be slightly higher at 6×6 for SI Pro-HOG. Given the very small size of the examples, this completely changes the resulting classification accuracy; precision is improved, but recall is degraded. At these cell-block dimensions, at least half of

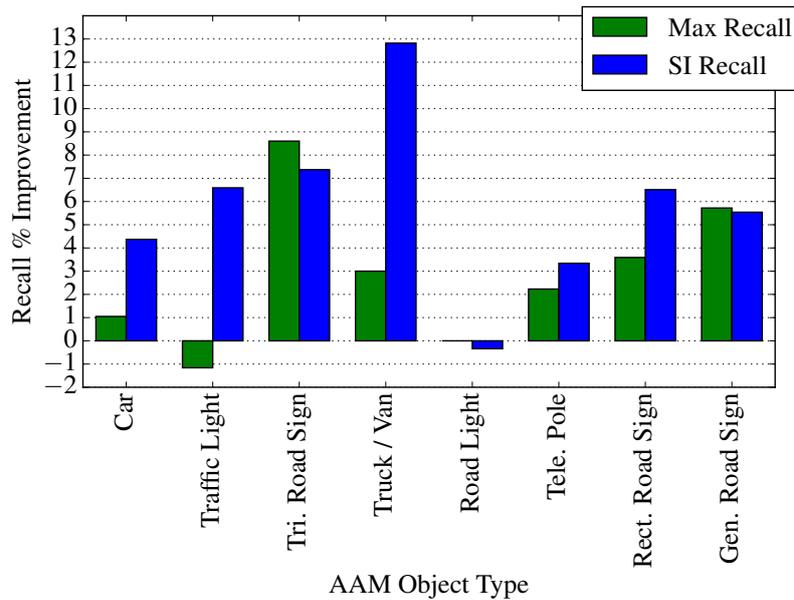


Figure 4-84: Percentage improvement in recall of the Pro-HOG (Scale Independent) and Pro-HOG (Max Resized) extractors from the Pro-HOG (Median Resized) extractor for object types from the AAM dataset (at object specific optimal cell-block dimensions).

the instances have their histograms constructed using only 9 pixels per cell.

Even though overall in figure 4-85, recall is not significantly improved, SI Pro-HOG still represents an improvement due to the fact that the extra processing required to resize the images is not needed to achieve a comparable level of accuracy to HOG.

The results in this section empirically corroborate the analysis of Dollar *et al.* (2014) and show that a reduction in feature encoding quality due to image scaling by a positive factor < 1 has tangible deleterious effects on classification accuracy. The results also justify the design of Pro-HOG in its ability to accurately extract features from images without first having to resize them.

By avoiding the extra processing required to resize the images needed by HOG, and only requiring feature descriptors with lower dimensionality for increased classification accuracy, Pro-HOG is algorithmically faster as well as space saving in memory.

Object (FV Dims)	FX (Image Dims.)	Precision	Recall	F1
Car (8100)	SI (native)	1.000	0.9038	0.9495
	Median (90×90)	0.9802	0.9519	0.9659
	Max (336×336)	0.9792	0.9038	0.9400
	HOG (336×336)	0.9804	0.9615	0.9709
Garbage Bin (2304)	SI (native)	0.9878	0.9918	0.9898
	Median (17×17)	1.0000	0.9918	0.9959
	Max (90×90)	0.9916	0.9713	0.9814
	HOG (90×90)	0.9873	0.9590	0.9730
Traffic Light (1296)	SI (native)	0.7981	0.7270	0.7609
	Median (34×34)	0.7857	0.6954	0.7378
	Max (322×322)	0.8514	0.7241	0.7826
	HOG (322×322)	0.8113	0.7040	0.7538
Parking Sign (1296)	SI (native)	0.9713	0.9476	0.9593
	Median (35×35)	0.9748	0.9476	0.9610
	Max (175×175)	0.9606	0.9371	0.9487
	HOG (175×175)	0.9478	0.8881	0.9170
Traffic Cone (1296)	SI (native)	0.9647	0.8039	0.8770
	Median (20×20)	0.9362	0.8627	0.8980
	Max (56×56)	0.9560	0.8529	0.9016
	HOG (56×56)	0.9773	0.8431	0.9053

Table 4.32: Earthmine dataset: comparing Pro-HOG’s scale independence versus image re-sizing using optimal cell-block dimensions.

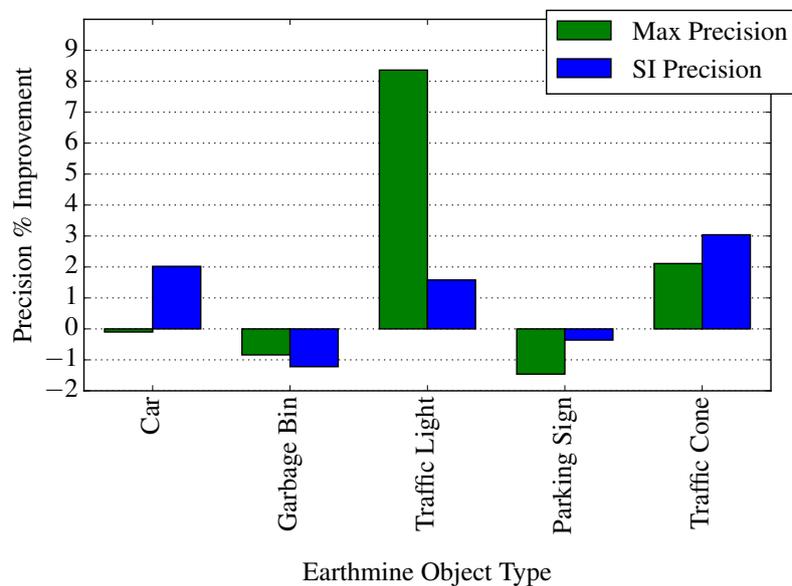


Figure 4-85: Percentage improvement in precision of the Pro-HOG (Scale Independent) and Pro-HOG (Max Resized) extractors from the Pro-HOG (Median Resized) extractor for object types from the Earthmine dataset (at object specific optimal cell-block dimensions).

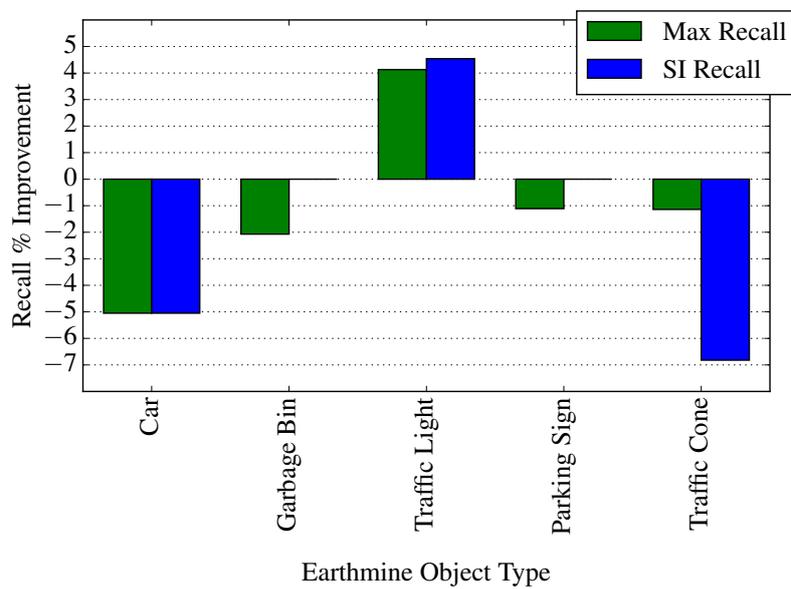


Figure 4-86: Percentage improvement in recall of the Pro-HOG (Scale Independent) and Pro-HOG (Max Resized) extractors from the Pro-HOG (Median Resized) extractor for object types from the Earthmine dataset (at object specific optimal cell-block dimensions).

4.5 Conclusion

This chapter introduced Pro-HOG – a new algorithm for the extraction of features from 2-D imagery based on Histograms of Oriented Gradients (HOG) by Dalal and Triggs (2005). The algorithm is novel in that it maintains a fixed sampling resolution for the generation of feature vectors irrespective of the dimensions of the subregion that the features are being extracted from. This allows region descriptors coming from two differently sized image regions to be directly compared without first scaling the extracts to identical dimensions.

Pro-HOG was compared against HOG in a rigorous suite of object classification experiments against three different datasets, and using two different types of input data (grey scale intensity images derived from colour images, and depth maps derived from stereoscopy and laser scanning). The initial hypothesis postulated that Pro-HOG would provide for improved classification accuracy over HOG due to its avoidance of image scaling and the possibility of attendant scaling artefacts.

While a strong indication of increased accuracy (especially recall) with Pro-HOG was seen in the AAM dataset – the largest of the three datasets tested, it is not possible to conclude that Pro-HOG is superior in general to HOG for the purposes of classification. This is because each object type was shown to be substantially different enough to require different settings of the feature extraction parameters (for both Pro-HOG and HOG) in order to obtain the optimal classification accuracy for the type. On the two smaller datasets tested – the Pascal VOC 2007 dataset (which does not include depth data) and the Earthmine dataset, no strong tendency towards increased classification accuracy using Pro-HOG was observed.

Section 4.2.4 tested the effects on classification accuracy of extract resizing. Resizing extracts in the case of the Pascal VOC 2007 “Person” dataset was shown to degrade recall accuracy. While this only constitutes evidence for one object type from a single dataset, depending on the nature of the objects/features being tested, using Pro-HOG instead of HOG may help to avoid this problem.

Even though no conclusive evidence was found for Pro-HOG having improved classification accuracy over HOG, the results do suggest that it is possible to fix the sampling resolution in the extraction mechanism prior to generating the feature vectors without incurring any loss in classification accuracy. This means that Pro-HOG has some computational advantages over HOG. A level of classification accuracy can be achieved with Pro-HOG that is at least as good (if not better) than HOG using shorter feature vectors comprised of a coarser sampling of the image data, hence this makes Pro-HOG more efficient in time and space because image

scaling is unnecessary, and less memory is needed to store the descriptors.

The classification experiments using the object types from the AAM dataset demonstrated that these advantages are especially evident with extracts having smaller dimensions. With smaller object extracts, the act of resizing them to fixed dimensions does not provide any informational advantage and so cannot aid classification. The resizing only serves to allow the feature extractor to encode the extract's features in a longer vector.

Due to its two stage implementation, two facets of the original HOG algorithm couldn't be implemented in Pro-HOG: cell resolution histogram interpolation and Gaussian smoothing at the same resolution. By its empirical validation against HOG, it was shown that the lack of these schemes in Pro-HOG does not confound classification accuracy.

Not explored in this chapter is the advantage offered by Pro-HOG's two stage algorithm when used in the context of object detection. With HOG, each extract (which may be any image subregion) must be independently resized before a descriptor for the extract can be generated. Performing this resizing operation in potentially thousands of different image locations is computationally expensive – especially where different subregions overlap. Because Pro-HOG processes the query image at a pixel resolution just once – storing the histogram orientations in an integral image, the task of generating the feature descriptors for arbitrarily sized subregions is made very much more efficient. An empirical assessment of this advantage especially in the context of detection where depth information is explicitly available is a candidate for future research.

4.5.1 Depth Based Features

As discussed in section 4.3, the Pro-HOG algorithm is implemented to represent input data using floating point values. It was designed this way to enable it to more accurately encode the differences in values in depth imagery. The reasoning behind this design decision was that more accurately encoded feature vectors from depth extracts would be more representative of changes in surface morphology in the object extracts and this would result in more accurate classifiers being learned. However, the results showed that while Pro-HOG performed better using depth based descriptors than with the intensity based descriptors, the size of the improvement in accuracy was not as great overall as that experienced by the original implementation of HOG when going from the intensity based to the depth based descriptors – even though the tested HOG implementation rounded the depth values meaning that small changes in surface morphology could not be represented. Overall, the effect of rounding the

depth values was to improve classification accuracy so that HOG performed at least as well as Pro-HOG with the depth based descriptors.

This result is best explained by the fact that when looking at the depth extracts of the positive instances from any of the object classes, they tend to be represented as solid planar surfaces expressing minimal variation in depth across their surfaces (which are shown orthogonally to the camera's viewpoint). For example, the orientation of road signs in the data is almost wholly face on to the camera – which is not unexpected given that the mobile mapping vehicle is road based and the vast majority of road sign examples are taken from the view of the mapping vehicle facing up the road in the direction of travel. On the other hand, the random negative extracts typically showed rapidly changing values in depth from horizontal surfaces such as the road, or from patches of vegetation (particularly in the case of the AAM dataset where the method of laser scanning allows for individual tree branches and the gaps between them to be imaged).

In the implementation of Pro-HOG tested here, the fact that the small changes in surface morphology were encoded in the feature vectors meant that the positive descriptors were *more likely* to be confused with the descriptors extracted from the negative examples. In the implementation of HOG tested here, these small changes in depth in the positive examples were ignored due to the rounding meaning that the positive descriptors were less likely to be confused for the negative descriptors. Because the negative examples expressed larger (on average) changes in depth – changes of greater magnitude than could be affected by rounding – HOG was able to encode the different classes using a much simpler characteristic: those descriptors encoding a lot of change in depth (the negative examples) versus those descriptors encoding very little change in depth over a significant portion of the descriptor (the positive examples).

This explanation implies two things. Firstly, that the improvement in accuracy realised overall by the depth based descriptors over the intensity based descriptors in the experiments in this chapter is best explained by the fact that any object of interest (whatever its type) is more likely to be present in those parts of the image where depth is changing only very minimally, if at all. Secondly, that depth data can provide excellent contextual information to quickly rule out those image areas that are unlikely to contain objects of interest – assuming that the objects of interest are never present at oblique angles. In this case, a complex feature vector extraction mechanism would not be required when undertaking object detection; a thresholded transform of the depth data showing change in depth (using a Sobel operator) would be sufficient. Detection and classification could then be carried out on the remaining parts of the image.

As a topic for possible further research, to more accurately ascertain the discriminative power of depth based features, it will be more instructive to compare the accuracy of Pro-HOG versus HOG when classifying over multiple object types instead of a single object type versus random negative extracts. This may show that Pro-HOG's encoding of the morphological changes in depth gives improved classification precision over the implementation of HOG tested here which rounds input values. It should be noted though, that this rounding issue is not intrinsic to the HOG algorithm *per se*, but is incidental to the version of HOG being evaluated here. There is no reason to think that Pro-HOG would be intrinsically better than a HOG implementation that used floating point values internally.

4.5.2 Limitations

In section 4.2.2, when evaluating the effect of setting the contrast gradient measurement to be sensitive or insensitive to the sign of the change, it was not possible to make a direct comparison with HOG for the case of gradient sign insensitivity because the OpenCV implementation does not allow this parameter to be changed. Because the results of these tests with Pro-HOG showed that gradient sign insensitivity may be preferred for recall, it could be the case that changing this parameter in HOG will result in slightly higher recall (and lower precision) values too, although the overall difference in precision and recall between the two settings of gradient sign sensitivity was found to be slight (and far less important for classification accuracy than changes to the histogram length or cell grid dimension parameters).

A second limitation of Pro-HOG is that it can only extract features from images having native pixel dimensions at least as large as the predefined cell grid dimensions. One workaround is to resize images that are below this limit up to the minimum allowed size – likely experiencing only a minimal degradation in classification / detection accuracy. Since the optimum cell grid dimensions are typically low (overall, 9×9 cells gave the best accuracy in testing), this is unlikely to present a problem in practice; objects represented at these dimensions are already too small to discern in most cases.

Chapter 5

Extending Randomised Hough Forests Using Depth Data

In previous chapters, feature descriptors encode the appearance characteristics of whole objects together with the relative spatial relationships of the parts of the objects using rigid descriptors. Chapter 4 shows that the explicit availability of depth means that these descriptors can be encoded using a scale invariant approach.

As noted in section 2.3, rigid descriptors are limited in their ability to accurately encode the salient features of an object type, especially when there is significant variation in how objects are represented in images. Section 2.3.2 introduced Implicit Shape Models (ISMs) which represent object types in such a way that many of the representational problems that afflict rigid descriptors can be mitigated. The Class Specific Hough Forest (CSHF) (Gall *et al.*, 2009) is a method of learning ISMs and comparing the generated models to features extracted from query images to detect the presence of objects of the modelled type.

In this chapter, three extensions to the CSHF that make use of explicitly available depth data are developed. The accuracy of object detection using the depth extended CSHF is evaluated and the results compared against the CSHF in its original implementation without the depth extensions. The depth extensions are shown to result in improved object detection accuracy over the original method. Importantly, the extensions do not require any algorithmic changes to the original CSHF algorithm but are effected by treating the inputs and outputs of the algorithm differently.

Randomised Hough forests have been used with 3-D data and depth imagery in several ways. Knopp *et al.* (2010) extended the SURF (Bay *et al.*, 2008) feature descriptor for 3-D object classification within a Hough forest based framework. Fully volumetric 3-D data have been

used to train object part dictionaries (or code books) so that parts at the incorrect scale can be rejected, improving the representative accuracy of the object models. This results in a reduced number of false positives at detection time (Sun *et al.*, 2010).

Using depth information, the scale of object parts has been used within a Hough forest framework to allow parts to vote for the locations of their comprising objects in three dimensions (as opposed to two dimensions as per the usual Hough forest scheme) (Salti *et al.*, 2010). However, this means that the voting space requires an extra dimension resulting in an increase in the sparsity of votes; an exponentially larger number of votes is required to maintain the same accuracy of detection. In the original evaluation of the CSHF, votes are also cast in scale space by resizing the query image and performing detection over each of the separate images (Gall *et al.*, 2009) which is inefficient and only allows for the detection of object parts at predefined scales.

In this chapter, part offsets are encoded using a 2-D vector, but the magnitude of the vector is related to the actual scale of each part in the image as found by directly measuring the depth at the location of the part. This *scale invariant* encoding of the part offset means that parts from training examples with different pixel dimensions can contribute accurate part offsets in the detection of an object at any scale in a query image.

The descriptors that encode the characteristics of a given part (defined as an image patch or subregion) are termed *patch descriptors*. The framework developed and evaluated in this chapter scales these patch descriptors depending upon the measured depth values from the query images. The magnitude of the part offsets is defined in terms of the scaled size of these local image regions. This means that salient part characteristics can be compared from local image patches extracted at different scales and the pixel dimensions of the part offset can be determined using the scale of the patches extracted from the query images. Scale proportion feature extraction techniques (such as that developed in chapter 4) are well suited to the generation of patch descriptors from local image regions having non-constant dimensions. In this chapter, the scaled extraction of features is applied in the context of a parts based object detector rather than a rigid template object classifier. The scaling of patch descriptors has been undertaken in other work (Wang *et al.*, 2012), but this used contextual patches rather than patches taken from the objects themselves. In this chapter, patch scaling is combined with scaled part offsets in an integrated approach to perform object detection and localisation.

To take advantage of the high accuracy depth information provided by the AAM dataset (see section 3.3) a simple depth based feature is developed to encode the local surface morphology of objects as they appear in depth images. This depth feature is compared against the set

of RGB based features used in the original CSHF evaluation (Gall *et al.*, 2009). It is shown that for some object types in the AAM dataset, the depth based feature which encodes local patches using only 10 scalar values can provide detection accuracy (in both recall and precision) that is at least as high as the original set of RGB based features which encode local patches using 8192 scalar values. This allows for order of magnitude speed ups in feature extraction and object detection, and significantly reduces memory requirements.

Hough forests have previously been combined with depth images to attempt to solve the problem of estimating human pose (Holt and Bowden, 2012). In training, the uncertainty in the spatial separation of an object’s parts has been considered in an attempt to improve the estimation of its centroid (Sun *et al.*, 2010; Wang *et al.*, 2012). In the framework presented in this chapter, depth values are used during detection to weight votes. It is assumed that the measured depth of individual parts is reasonably consistent across the projected face of an object. This assumption is leveraged to decrease the weight of votes from source part locations where the disparity in depth between the location of the part and the voting location is large. It is hypothesised that this will decrease the number of false positive votes, thereby increasing detection precision at a given recall rate.

This chapter is organised as follows. Section 5.1 gives a brief overview of the CSHF framework as developed for 2-D object detection in Gall *et al.* (2009). Section 5.2 describes the design and implementation of the depth-based extensions to the CSHF object detection framework and how they are used during the training (forest generation) and detection (vote aggregation) phases. Section 5.3.2 describes how the depth data in the Earthmine dataset are adjusted to account for the lower quality depth information. These adjustments are made according to knowledge about the kind of objects being detected, as well as how the Earthmine data were captured. Section 5.3 describes the experimental methodology used to compare the results garnered from the original CSHF detector against the Earthmine and AAM datasets with the CSHF detector using the new depth extensions against these data. Section 5.4 presents the outcomes of the experiments and provides an analysis of the efficacy of the depth based extensions. Finally, section 5.5 concludes this chapter and describes areas for future research.

5.1 Class Specific Hough Forests

This section gives an overview of the Class Specific Hough Forest (CSHF) algorithm as presented in Gall *et al.* (2009).

Each tree in the CSHF specifies a set of leaf nodes that describe the parts of a particular class of object and each tree is constructed to give a slightly different representation of a model that encodes the appearance and location of the parts for the object type. The set of leaf nodes in each tree represents a dictionary (or codebook) of implicitly learned parts. As described in section 2.3.2, a boosted approach to classification using these differing decision trees (termed a *random forest*) means that many of these slightly different models can be aggregated together, and the classification results of all of them used to make a final determination as to the location of the objects in query images. This has been shown to result in improved generalisation and stability than for using a single tree (Amit and Geman, 1997). For Hough forests, this amounts to each tree making its own slightly different set of votes in Hough space given the same input data. The essential difference between the trees is the randomisation of decisions during tree generation – the algorithms used to generate each tree are the same and so the description of the algorithms that follows concerns a single tree.

A ground-truthed dataset for a single object type consists of examples of the object type delimited with bounding boxes. Features extracted from small image subregions within these bounding boxes are used to describe local parts of objects. Small fixed size regions are used because actual object “parts” are heterogeneous across object types – varying in their shape and extents. Small fixed size rectangular regions are simpler to compute and are less variable in their appearance than the objects they describe. The variability of these smaller parts is simpler to model than the variability of their comprising objects and it is this notion that underpins the basis for modelling the complexity of an object by a configuration of its many simpler parts. The specific configuration is given by a generated tree which partitions these parts (as encoded by patch descriptors) having similar “appearance” according to a similarity metric calculated over the descriptors, as well as by similar offsets from the reference points of the objects the parts are taken from. In the original CSHF algorithm, the object centroid denotes the part reference point.

A patch descriptor extracted over a single local image region can be comprised of one or more independently acquired feature vectors of different dimensions; each feature measuring different characteristics of the region. The concatenation of the feature vectors comprises the complete descriptor for the image region (which is interpreted to be a part of an object). Objects classes are modelled by thousands of such patch descriptors, each extracted from a random location inside a bounding box delimiting an example instance of the class. The offset of each patch descriptor to a standard reference point relative to the bounding box of the example it was generated from is stored along with the patch descriptor. The patch descriptor and the 2-D part offset vector are used during the training phase to generate each tree of the CSHF.

5.1.1 Training (Tree Generation)

The decision trees are binary *i.e.* all nodes have exactly two children apart from the leaf nodes that have none. A node represents a decision point that evaluates either the appearance of a part (its descriptor), or the part offset (the 2-D vector to the centre of the comprising object instance). The objective of training the trees is to group together patch descriptors that are similar in appearance (classification) and that also have similar offsets to the centres of their comprising objects (regression). Once trained, the tree can be presented with new patch descriptors extracted from query images. New patch descriptors are evaluated along a path of nodes starting at the root node from parent to child until arriving at the leaf node which determines how votes for the object centroid should be cast given the patch extract.

To train a random forest, each tree is trained on a random subset of the original training dataset, with members of the subset sampled (with replacement) from the full training dataset. This means that some instances may be included in the random training set more than once which has the effect of weighting the decision tree more toward those examples. Within the random forest framework, these differences when aggregated over multiple decision trees help to cancel one another out – improving the overall accuracy of the votes (Amit and Geman, 1997). Each tree is trained independently, and so each tree can be trained concurrently – decreasing the actual amount of time needed to train the entire forest.

After selecting the random subset of M training instances from the total set of ground-truthed examples of the object class, each instance is parsed in turn to generate N patch extracts sampled from random locations from within its bounding box. Unlike earlier Hough forest detection frameworks which generate codebooks of object parts based on detected interest points (Leibe *et al.*, 2004, 2006, 2007), interest point detectors are not used with the CSHF. This means that object parts can be more densely sampled, providing more information as to the possible location of an object given the same amount of data. This also means that any *a priori* assumptions about the usefulness of certain image features is avoided and the utility of the different patch descriptors is learned by the decision trees. Dense sampling also improves the robustness of the detector to noise and inaccuracies in the training data (Gall *et al.*, 2009).

Each patch extract is defined as a three tuple $\epsilon = (c, \mathbf{q}, \mathbf{v})$ where $c \in \{0, 1\}$ specifies whether the extract is from a negative (0) or a positive (1) example, \mathbf{q} is the descriptor for the local patch, and \mathbf{v} is the offset vector to the centre of the ground-truthed example. For negative extracts, the value of \mathbf{v} is left undefined since it is ignored. Equal sets of positive \mathcal{E}_1 and negative \mathcal{E}_0 patch extracts are generated and these are supplied to a tree for training.

Training proceeds by recursively partitioning the set of training positive and negative extracts received by each node according to either a classification metric based on the appearance of each patch extract’s descriptor, or a regression metric based on the similarity of the offset vectors in the positive patch set. Recursion ends if one of the following four conditions are met.

1. The tree’s maximum depth is reached.
2. Only negative patch extracts remain in the training set ($|\mathcal{E}_1| = 0$).
3. The training set’s minimum size has been reached.
4. It is not possible to find a binary partition of the training set using the given partition function such that both of the partition sets are not empty within a fixed number of tries.

The maximum depth of a tree is set to 15 and the minimum allowed number of patches per node is set to 20. The choice of whether to partition a node’s training extracts according to a classification or a regression metric is made randomly unless less than 5% of the training patches are negative, or the node is within three levels of the maximum tree depth in which case regression is used.

For each node, a partition of the training patches is sought over a fixed number of iterations (20,000 by default) that maximises the metric for the classification or the regression function. Each partition of the data is generated randomly by selecting two random indices i and j from the k^{th} randomly selected feature vector of the patch descriptors in the training set. This gives two, two dimensional indices into each training patch descriptor \mathbf{q} that are related to the same feature. For each positive and negative patch descriptor \mathbf{q} , the difference between these indexed feature values is found as $\delta = \mathbf{q}_{ki} - \mathbf{q}_{kj}$. From these comparisons, the smallest δ_{\min} and the largest δ_{\max} differences over all of the training patches are found. A threshold value $\tau \in [\delta_{\min}, \delta_{\max}]$ is then randomly generated, and the patches are partitioned into one of two prospective training sets for the current node’s children; set \mathcal{A} if $\delta < \tau$, or set \mathcal{B} if $\delta \geq \tau$. It is over these prospective sets \mathcal{A} and \mathcal{B} that the classification and regression metrics are calculated.

If classification is used, then the metric is calculated in terms of the entropy of the prospective training sets. Letting \mathcal{A} be the prospective training set for the left child node, and \mathcal{B} be the prospective training set for the right child node, the classification metric is defined according

to the combined entropy of the two sets U_1 given by

$$\text{entropy}(x) = -x \log x - (1 - x) \log(1 - x) \quad (5.1)$$

$$U_1 = |\mathcal{A}| \text{entropy} \left(\frac{|\{\mathcal{A} | c = 1\}|}{|\mathcal{A}|} \right) + |\mathcal{B}| \text{entropy} \left(\frac{|\{\mathcal{B} | c = 1\}|}{|\mathcal{B}|} \right) \quad (5.2)$$

If regression is used to partition the patch extracts, the metric calculated is the variance between the part offset vectors over the positive patch extracts for the two prospective training sets. This metric U_2 is defined as

$$U_2 = \sum_{i \in \mathcal{A} | c=1}^{\mathcal{A}} (\mathbf{v}_i - \mathbf{v}_{\mathcal{A}})^2 + \sum_{i \in \mathcal{B} | c=1}^{\mathcal{B}} (\mathbf{v}_i - \mathbf{v}_{\mathcal{B}})^2 \quad (5.3)$$

where $\mathbf{v}_{\{\mathcal{A}|\mathcal{B}\}}$ is the mean offset vector for the positive extracts in prospective training set $\{\mathcal{A}|\mathcal{B}\}$. Importantly, because the regression metric favours part offset distributions having a single mode, parts that are similar in their appearance but have very dissimilar offsets (because they are extracted from different relative locations in their comprising objects) are grouped into separate sets. This increases the accuracy with which votes are eventually cast because it decreases the problem of parts having too much variability in their allowed positions relative to the object centroid.

The prospective training sets that are selected are those that minimise the selected metric (either U_1 or U_2) and the parameters that provide this partition of the training extracts i, j, k , and τ are stored in the node. Training recursively propagates to the left and right child nodes supplying the partitioned datasets \mathcal{A} and \mathcal{B} respectively as the new training sets for the nodes.

The randomisation of whether to partition each node’s training data based on classification or regression, as well as the randomisation of the parameters to these functions, further helps to ensure that each tree in the forest is substantially different. Once training is complete for a tree, each of its leaf nodes represents a single entry in the ISM codebook comprising patches having similar appearance (descriptors) and similar part offsets. The values stored in each leaf node are the proportion of positive patch extracts, and the distribution of offset vectors of these positive patch extracts.

5.1.2 Detection (Vote Aggregation)

The detection of objects using a CSHF proceeds by extracting patch descriptors from each location \mathbf{l} in the query image \mathcal{Q} . Letting f be the patch descriptor extractor function, each patch descriptor $\mathbf{q} = f(\mathcal{Q}, \mathbf{l})$ is supplied to each tree’s initial node (its root node) in the random forest. The task of detection is to find each tree’s leaf node that represents the codebook entry that is most similar to \mathbf{q} . To achieve this, \mathbf{q} is evaluated using the stored partition parameters i, j, k , and τ stored in each node during training. If $\mathbf{q}_{ki} - \mathbf{q}_{kj} < \tau$, \mathbf{q} is passed to the left child node, otherwise \mathbf{q} is passed to the right child node.

This process continues for each subsequent node until the tree’s matching leaf node, \mathcal{L} is found. Determining the correspondence of \mathbf{q} with a tree’s leaf node is efficient because the comparison is base 2 logarithmic in the number of leaf nodes stored in the tree.

Let \mathcal{H} be a 2-D matrix that specifies the Hough image (the object vote space) having the same pixel dimensions as \mathcal{Q} . Let the boolean random variable $E_{\mathbf{x}}$ denote the existence of an object at location \mathbf{x} in \mathcal{Q} . If \mathbf{q} is generated from a local patch that is outside of the bounding box of an object centred at \mathbf{x} , then the probability of there being an object at \mathbf{x} is independent of \mathbf{q} . That is,

$$P(E_{\mathbf{x}} | \mathbf{q}) = P(E_{\mathbf{x}}) \quad (5.4)$$

and no votes are cast in \mathcal{H} . While a scene’s context is useful for inferring the presence of an object, this aspect of detection is not dealt with explicitly by the ISM modelled by the CSHF (notwithstanding the extent to which background scene elements are inevitably captured inside of most training example bounding boxes and contribute as “pseudo parts” to the detection of those objects). However, the CSHF assumes that all \mathbf{q} are possibly located within the bounding box of an object of the modelled type. This implies that every patch extracted at position \mathbf{l} is from an instance of the object type modelled by the CSHF meaning that the boolean random variable $C_{\mathbf{l}}$ is true where C is the positive class label indicator. This allows the probability of the existence of an object at \mathbf{x} to be written as

$$P(E_{\mathbf{x}} | \mathbf{q}) = P(E_{\mathbf{x}}, C_{\mathbf{l}} | \mathbf{q}) \quad (5.5)$$

and by the general product rule of probability

$$P(E_{\mathbf{x}}, C_{\mathbf{l}} | \mathbf{q}) = P(E_{\mathbf{x}} | C_{\mathbf{l}}, \mathbf{q})P(C_{\mathbf{l}} | \mathbf{q}). \quad (5.6)$$

The factor $P(C_{\mathbf{l}} | \mathbf{q})$ in equation 5.6 is found simply as the proportion of the positive patch extracts stored in \mathcal{L} . The first factor can be estimated using *Parzen-Window* density estimation by comparing the offset vector $\mathbf{d} = \mathbf{l} - \mathbf{x}$ against every training offset $\mathbf{v} \in \mathcal{L}$. This gives

a probability estimate for a single tree of

$$P(C_l | \mathbf{q}) = \frac{|\{\epsilon \in \mathcal{L} \mid c = 1\}|}{|\mathcal{L}|} \quad (5.7)$$

$$P(E_x | C_l, \mathbf{q}) = \frac{1}{|\{\epsilon \in \mathcal{L} \mid c = 1\}|} \sum_{\mathbf{v} \in \mathcal{L}} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|\mathbf{x} - \mathbf{l} - \mathbf{v}\|^2}{2\sigma^2}\right) \quad (5.8)$$

$$P(E_x, C_l | \mathbf{q}) = \frac{1}{|\mathcal{L}|} \sum_{\mathbf{v} \in \mathcal{L}} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|\mathbf{x} - \mathbf{l} - \mathbf{v}\|^2}{2\sigma^2}\right) \quad (5.9)$$

$$(5.10)$$

where $\sigma^2 \mathbf{I}_{2 \times 2}$ is the covariance determined by the offset vectors stored in \mathcal{L} . The final probability over the entire random forest \mathcal{R} is then simply the mean of equation 5.9 calculated over every tree $\mathcal{T} \in \mathcal{R}$ derived as

$$P(E_x, C_l | \mathbf{q}; \mathcal{R}) = \frac{1}{|\mathcal{R}|} \sum_{\mathcal{T} \in \mathcal{R}} P(E_x, C_l | \mathbf{q}; \mathcal{T}). \quad (5.11)$$

The votes for the object at position \mathbf{x} are given by the value at position \mathbf{x} in \mathcal{H} which is generated as a simple sum of the independent part probabilities at each \mathbf{l} in \mathcal{Q} . Although aggregating the individual part probabilities in this way discards the probabilistic interpretation, it allows \mathcal{H} to express the relative confidence of object centroid occurrences where higher values denote greater confidence. The locations of the highest of these confidence scores can then be extracted as a set of object location hypotheses.

Unfortunately, the preceding derivation of $P(E_x, C_l | \mathbf{q})$ cannot be efficiently undertaken due to the computational complexity of calculating equation 5.9 for every extracted \mathbf{q} . As a substitute, the proportion of positive patches in each tree's matched leaf node (equation 5.7) is added directly to the values in \mathcal{H} at the locations $\{\mathbf{v} - \mathbf{l} \mid \mathbf{v} \in \mathcal{L}\}$. The final Hough image is then Gaussian filtered using a fixed size kernel.

In the original implementation, Gall *et al.* (2009) deal with scale variations by generating n separate Hough images after resizing the query image with scale factors s_1, s_2, \dots, s_n . These scaled images are set into a 3-D scale space frustum and Gaussian filtration is performed over the scale dimension to find estimates in a 3-D Hough space of the object hypotheses. Practically, this step is computationally costly and only a coarse estimation of an object's scale can be derived. The accuracy also depends upon having a large amount of training examples at all of the different scales at which new objects are expected to occur in the query images.

5.2 Depth Based Extensions

This section describes the depth-based extensions to the CSHF object detection framework and how they are incorporated into the training (tree generation) and detection (vote aggregation) phases.

5.2.1 Patch / Offset Scaling

Objects of the same type are assumed to have very similar real world dimensions. For example, most cars are about three to four metres in length. In the original formulation of the CSHF, the dimensions of each part patch are set *a priori* as fixed pixel dimensions in accordance with the requirements of the feature extraction algorithm – irrespective of the pixel dimensions of the training examples. The characteristics of these patches therefore depends upon the scale of the examples they are extracted from. Without explicit scale information, this diversity of scale among the training data helps to train the CSHF in such a way that no particular scale is emphasised (as long as dimensions of the training examples are representative of the scale of the expected instances in the query images). Since there is inevitable disparity between the representative object patches in the training data and the representative areas in the patches extracted from the query images, object detection is carried out over different image scales and the resulting Hough vote maps are aggregated (Gall *et al.*, 2009). This scheme requires a lot of training data to gain accurate results because there are a lot of false positive part patch detections at incorrect scales.

Using depth information and a feature extraction algorithm that can operate over arbitrarily sized image patches to generate constant size feature vectors, part patches that are more representative of object parts can be used to train the CSHF because they express invariance to changes in scale. The Pro-HOG feature extractor of chapter 4 was developed with such scale invariance in mind. However, given depth information, any feature extraction method can be used since the input image subregions (which scale in dimensions according to measured depth) can be rescaled to the pixel dimensions required by the feature extraction algorithm being used. In this chapter, this allows the same features to be extracted as in the original evaluation of the CSHF (Gall *et al.*, 2009) so that comparing between the original and the CSHF with depth extensions enabled is not unduly confounded.

Each ground-truthed training example has a boundary defined by a closely fitting rectangle (or bounding box). The bounding box’s centre is used as the nominal reference point, but any point that is given in terms of the object’s bounding box dimensions can be used. Because

the objects under evaluation in this chapter all have bases that are incident with the ground plane, the bottom middle of the object’s bounding box is used as the reference point. The reason for this is related entirely to the method of bounding box detection that is used, which relies upon the accurate measurement of the object’s depth. It is not guaranteed that a detected object has a valid surface in depth at its centre and so a more reliable location from which to measure the object’s range from the camera plane is at its base. Even if the object meets the ground plane at only one narrow point, the height of the object’s bounding box will ensure that most points along the base of the object’s bounding box will be very nearly the same in depth. For example, a traffic light’s pole may not be aligned along the vertical centre of its bounding box making any determination of its depth unreliable if its depth is measured at centre of the bounding box.

Let the training dataset for object type c be given as a set of tuples $T_c = \{(\mathcal{I}, \mathbf{g})\}$ where \mathcal{I} is the coregistered image and depth data within which the example is located, and \mathbf{g} is a four dimensional vector giving the pixel position and dimensions of the example’s bounding box.

To determine the dimensions of each randomly located part patch within the bounding box of an example in T_c , it is first necessary to set the real world 2-D dimensions of a part patch $\mathbf{\Gamma}_c$. This is a two dimensional vector specifying the horizontal and vertical dimensions of a patch in the same units as the depth values in the query data (*e.g.* metres). The dimensions of a part patch can be specific to the object type being modelled by the CSHF. For any particular object type and feature extraction method, an optimal value of $\mathbf{\Gamma}_c$ can be found using cross-validation.

Let $\text{depth}(\mathcal{I}, \mathbf{x})$ give the depth at point \mathbf{x} in \mathcal{I} and let λ be the known focal length in pixels of the imaging camera. The pixel dimensions $\boldsymbol{\gamma}$ of a part patch located at position \mathbf{l} in $\{\mathcal{I}, \mathbf{g}\}$ can be found by similar triangles as:

$$\boldsymbol{\gamma}_{\mathbf{l}} = \frac{\lambda}{\text{depth}(\mathcal{I}, \mathbf{l})} \mathbf{\Gamma}_c \quad (5.12)$$

This method requires that depth data be available with the training data and that it be precise enough to describe the shape of the objects in depth. If the training data do not contain depth information, an alternative method using the ratio between the fixed real world patch dimensions $\mathbf{\Gamma}_c$ and the estimated average real world dimensions of the object type $\hat{\mathbf{m}}_c$ can be used. This ratio is used to factor the pixel dimensions of the example’s bounding box

to derive an estimate of a patch’s pixel dimensions $\hat{\gamma}$ as

$$\hat{\gamma} = \left\langle \frac{\Gamma_h g_h}{\hat{m}_h}, \frac{\Gamma_v g_v}{\hat{m}_v} \right\rangle \quad (5.13)$$

where $\mathbf{\Gamma}_c = \langle \Gamma_h, \Gamma_v \rangle$, $\hat{\mathbf{m}}_c = \langle \hat{m}_h, \hat{m}_v \rangle$, and $\mathbf{g} = \langle g_x, g_y, g_h, g_v \rangle$.

The advantage of this approach is that the training data do not require any depth data. The disadvantage is that locations within an example’s bounding box that do not fall upon the surface of the object itself (*i.e.* the location is outside the true extent of the object but within the example’s bounding box) will have pixel dimensions for the “part” at that location calculated that are not commensurate with the *actual* depth of the part at that location.

While the derivation of the patch pixel dimensions γ in equation 5.12 is parameterised with the location of the patch \mathbf{l} which makes the measurement dependent upon the coregistered depth, the derivation of γ in equation 5.13 is an estimate, both because it relies upon an estimation of the object’s true dimensions, and because it is independent of the location of the patch. This means that the pixel dimensions of all of the patch extracts for a single training exemplar are the same using this method. During detection (vote aggregation), equation 5.12 is used to determine the appropriate apparent size of a part patch for a known class c and extraction location \mathbf{l} .

Offset Scaling

The idea behind scaling the offset vectors of parts to their comprising object reference points is that an object presents similar values in depth over its 2-D projected surface. In other words, pairs of points taken from unrestricted random locations in a depth image will express much greater variability in depth on average than pairs of points that are randomly sampled from the area delimited by the 2-D projection of a single object. More definitely, the absolute difference in depth between any part of an object and a reference point on the object should be small.

The part offset vectors in the original CSHF implementation are set in terms of horizontal and vertical pixel distances. Without depth information, this is reasonable since parts are then used to vote for an object *at a particular scale*. Without depth, a part descriptor extracted from a smaller training example cannot be used to vote for the presence of an object in a query image at a larger scale for two reasons. Firstly, the parts of the larger scaled object are concordantly larger also and so any extracted patch descriptor from the query image will represent the characteristics of a part that is proportionately smaller (because fixed

pixel dimensions are used to generate the patch descriptors). This means that the extracted query patch will likely not match the appearance of a patch extracted at the same relative location from any of the training examples unless those training examples have the same pixel dimensions. Secondly, even if patches extracted from the query image do match the training patch descriptors stored in the CSHF, the scale difference means that the encoded offset vectors in the training patches will incorrectly cause votes to be cast at locations in Hough space where the query image object’s reference point *isn’t*, amounting to many false positive detections of object reference points.

The assumption that object parts are similar in depth implies that the distances between parts should be scaled according to measured depth. Since objects of the same type are assumed to have similar real world dimensions, and parts taken from the same spatially relative locations of an object should have similar scaled pixel dimensions, a part’s offset vector can be encoded as the ratio of its distance (in pixels) to the object reference point \mathbf{r} with the dimensions of the part patch γ at \mathbf{l} calculated according to equation 5.12 (or equation 5.13 if depth data are not available). The object reference point \mathbf{r} is set here to be the middle base of each example’s bounding box. Given a part located at \mathbf{l} in training image \mathcal{I} , the extract offset \mathbf{v} to the reference point of the object specified by bounding box \mathbf{g} is found as

$$\mathbf{r} = \langle g_x + \frac{g_w}{2}, g_y + g_h \rangle \quad (5.14)$$

$$\mathbf{v} = \langle \frac{r_x - l_x}{\gamma_w}, \frac{r_y - l_y}{\gamma_h} \rangle \quad (5.15)$$

Because the formulation of the offset vector in equation 5.15 is unitless, this method of encoding the offset vectors can be used regardless of whether patches use fixed pixel dimensions (as in the original CSHF) or have their dimensions calculated using equation 5.12. As long as the same patch dimension derivation method is used during both the training and the detection phases, this method of encoding offsets as a proportion of the patch dimensions allows the correct offset magnitudes to be derived at detection time.

The part offset vectors \mathbf{v} point *toward* the object reference points in this formulation. The original CSHF algorithm (Gall *et al.*, 2009) has the offsets pointing toward the object parts but this difference is irrelevant as long as the calculations are consistent throughout and that ultimately the voting vectors are calculated to give the location of the object reference points relative to the extracted patches.

The offset vectors calculated using equation 5.15 are unitless. This means that during training, their magnitudes are standardised for parts coming from the same relative position in an object – even if the training examples those parts are extracted from differ in scale. The ISM

trained in the CSHF is then effectively a scale independent model of the object type.

At detection time, the pixel dimensions of a patch in the query image (determined using equation 5.12) can be used to calculate the offset vectors in pixels giving the correct scaled location of a potential object’s reference point relative to the patch. Given a patch located at \mathbf{l} in query image \mathcal{Q} , the patch dimensions γ at \mathbf{l} can be found according to equation 5.12. The absolute pixel location to vote for the presence of a potential object’s reference point \mathbf{u} can then be calculated simply as

$$\mathbf{u} = \mathbf{l} + \langle \gamma_w v_x, \gamma_h v_y \rangle \quad (5.16)$$

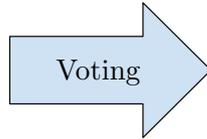
for every positive patch descriptor \mathbf{v} matched by the CSHF given the query patch descriptor extracted from $\gamma_{\mathbf{l}}$.

It is hypothesised that this method of deriving patch dimensions and encoding offset vectors using the provided depth information, should improve the accuracy with which votes can be cast for the object reference points, improving precision at a given recall rate. The scale independent encoding of offset vectors in the leaf nodes of the CSHF should also improve recall. This is because the regression metric as used in the CSHF is sensitive to the magnitudes of the offset vectors; positive training patches can be partitioned based on scale (leading to the fact that the original implementation of the CSHF can only accurately cast votes for objects at a particular scale). Since the magnitude of the unitless offset vectors will be in agreement (for parts extracted from the same relative locations in an object), patch descriptors having similar appearance are more likely to remain clustered together during the training phase. A given part is then represented by a larger number of positive patch descriptors resulting in a more representative encoding of the part’s variation. More reliable matching of new patch descriptors taken from query images should therefore be possible since more decisions to partition the training data have been taken in terms of the appearance characteristics of the training patch descriptors rather than the associated offset regression metrics. However, the improved patch appearance recall accuracy will not be to the detriment of the CSHF’s ability to group together similar part offset vectors since they are now scale independent.

Figures 5-1 and 5-2 demonstrate the differences between fixed magnitude part offsets as used in the original implementation of the CSHF (Gall *et al.*, 2009), and scaled magnitude part offsets used herein where the distance of a part to its voting location is scaled in proportion to its real world distance from the camera plane.

Figure 5-1 represents the original CSHF implementation. In this case, the training patch extracted from the larger scaled training instance of a traffic light has associated with it

Fixed part offsets



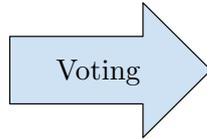
Low accuracy votes for object centroids

Figure 5-1: Fixed magnitude offsets cause votes to be cast inaccurately when matching object parts with different scales.

an offset vector of fixed length indicating the object centroid as the reference point. The magnitude of this vector is determined by the pixel dimensions of the training example. When matching parts are discovered in the query image, the offset vector is used at its original magnitude without any scaling. This causes a vote to be cast some distance from the centres of objects that aren't at the same scale as the training example. In the query image on the right of figure 5-1, the patches that are detected are false positives in that they incorrectly match the wrong parts of the traffic light poles. This is one source of error in the estimated object vote positions. The second source of error is due to the magnitude of the offset vector which is too large given the smaller scales of the detected objects in relation to the training example.

In figure 5-2, the same patch descriptor from the training data is used, but this time the offset vector dimensions are encoded as a proportion of the patch dimensions. When matching parts are detected in the query image, the scaled pixel dimensions of the corresponding patches determine the magnitudes of the respective offset vectors. In the query image on the right of figure 5-2, even though the matched parts are still false positive detections, the scaling of their offset vectors means that the cast votes are closer to the reference points of the respective objects than in figure 5-1. The errors are still present, but they are not as pronounced. The aggregate result is to reduce the spatial extents of voting errors for detected objects – potentially improving precision.

Scaled part offsets



Improved voting accuracy for object centroids

Figure 5-2: The pixel dimensions of scaled offsets are found according to the dimensions of the detected parts. This causes votes to be cast closer to the true locations of the object reference points.

5.2.2 Depth Surface Features

For a baseline patch descriptor extraction method, the same set of features as defined by Gall *et al.* (2009) are used. This feature extraction method was not designed with scale invariance in mind since the process accepts image subregions having fixed pixel dimensions. Its use here shows that a feature extraction method that was not specifically designed for scaled patch descriptor extraction can still be used within the depth extended CSHF framework presented in this chapter.

Features derived from the depth maps can be used together with (or instead of) features extracted from patches of the corresponding colour images. Because depth information is a lot less susceptible to natural imaging variations (such as lighting issues), the use of a depth based descriptor may allow for a more consistent encoding of local surface morphology from spatially respective object parts than may be possible using colour based patch descriptors alone. To this end, a simple descriptor was designed to encode changes in depth over local surface regions of an object.

Colour Image Based Features

Each extracted image patch is first scaled to fixed dimensions of 16×16 pixels. Each descriptor encodes 32 different feature types giving a 8192 dimensioned local descriptor (a feature value is extracted from each of the 256 pixels in a patch). The features include the three *Lab* colour space channels, the absolute values of the horizontal and vertical derivatives $|\delta/\delta_x|$ and $|\delta/\delta_y|$, the absolute values of the second degree derivatives $|\delta^2/\delta_x^2|$ and $|\delta^2/\delta_y^2|$, together with nine simple HOG-like channels (Dalal and Triggs, 2005). These 16 features are parsed with min and max filters in 5×5 pixel neighbourhoods to generate the 32 separate features. Min and max filtering helps to reduce the degree of variability between parts (as they are represented by the descriptor) taken from similar locations for a given object type.

Depth Map Based Features

The depth descriptor samples points at relative locations in arbitrarily sized square patches to allow for the scaling of patch dimensions. For a square patch, five points are considered: the four corners of the square, which are referred to clockwise from the top left as $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$, and \mathbf{p}_3 , and the centre of the square, \mathbf{p}_4 . The depth value at each of these points is compared to the depth value at every other point. Only the absolute difference in depth between each pair of points is encoded and so the ordering of the points in each pair is irrelevant. For n points, this gives $\frac{n(n-1)}{2}$ depth difference values. For $n = 5$, the results can therefore be stored in a vector having only 10 dimensions. Larger numbers of points in different relative positions were tried ($n = 9, n = 11$, and $n = 13$) but in limited testing, none of these improved object detection accuracy over the five point configuration. For any two points $\mathbf{p}_i, \mathbf{p}_j$ in image \mathcal{I} , the depth difference feature value extraction function ddfx is defined as

$$\text{ddfx}(\mathbf{p}_i, \mathbf{p}_j, \alpha) = \frac{1}{2} + \text{sign}(\delta) \frac{\min(|\delta|, \alpha)}{2\alpha} \quad (5.17)$$

where

$$\delta = \text{depth}(\mathcal{I}, \mathbf{p}_i) - \text{depth}(\mathcal{I}, \mathbf{p}_j) \quad (5.18)$$

$$\text{sign}(\delta) = \begin{cases} 1 & \text{for } \delta \geq 0 \\ -1 & \text{for } \delta < 0 \end{cases} \quad (5.19)$$

Parameter α specifies a positive value giving the sensitivity in depth (in metres) being encoded. The resulting range of ddfx is $[0, 1]$. Differences in depth greater than α therefore cause ddfx to take on its maximum value of one. Parameter α should be set according to the expected variation in depth over a scaled patch region.

5.2.3 Depth Weighting

The assumption that the depth of object parts is similar across the projected surface of an object is further exploited. During vote aggregation, the estimated locations of the potential object reference points are calculated according to equation 5.16. For a single vote, if the depth at the estimated object reference point is dissimilar to the depth at the location of the part, the confidence in the vote should be lower. Inversely weighting the strength of votes by the absolute difference in depth between the part and the estimated object reference point accomplishes this.

In the standard CSHF algorithm, a part’s vote for its object reference point is weighted by the proportion of positive parts in a tree’s matched leaf node \mathcal{L} as per equation 5.7. Given the classification of a patch descriptor located at \mathbf{l} in query image \mathcal{Q} , the “strength” of the vote of each offset $\mathbf{v} \in \mathcal{L}$ is given by

$$h(\mathbf{l}, \mathbf{u}) = \frac{|\{\epsilon \in \mathcal{L} \mid c = 1\}|}{|\mathcal{L}| |\text{depth}(\mathcal{Q}, \mathbf{l}) - \text{depth}(\mathcal{Q}, \mathbf{u})|} \quad (5.20)$$

where \mathbf{u} is found for each $\mathbf{v} \in \mathcal{L}$ according to equation 5.16. This particular weighting was selected because it gave the best results during testing. Functions that were tried but were found to be less effective included exponentiation of the absolute difference in depth, as well as the squared difference in depth. As in the original algorithm, each newly weighed vote $h(\mathbf{l}, \mathbf{u})$ is aggregated at position \mathbf{u} in \mathcal{H} .

By this method, not every offset vector $\mathbf{v} \in \mathcal{L}$ contributes an equal voting magnitude because the offsets all differ slightly even though partitioning via the CSHF’s regression metric causes similar offsets to be grouped together in the leaf nodes. The individual vote weighting thus gives greater emphasis to the parts stored in a leaf node that are more likely to comprise the object. It is hypothesised that this should reduce the degree of false positive responses by narrowing the spatial extents of the modes in Hough space that indicate potential object detections.

5.3 Experimental Method

Experiments were conducted using the AAM and Earthmine datasets with the object types listed in table 5.1. Each image contains at least one example of one of the object types being evaluated. The third column shows the total number of examples in the object dataset. The fourth column specifies the number of images the examples are taken from. The object

categories are the same as those used for evaluation in chapter 4 with the exception of the “Car” and “Person” objects from the Pascal VOC 2007 dataset because this has no associated depth imagery. Also excluded is the AAM “Generic Road Sign” object category because it contains overlapping ground-truth bounding boxes and the method of evaluating detection accuracy requires that ground-truth bounding boxes do not overlap (see section 3.5). Only examples within 100 metres of the camera plane are considered. This requires excluding a small number of examples from the Earthmine “Traffic Light” and “Parking Sign” object datasets due to their distance.

Dataset	Object	Count	Images
Earthmine	Car	52	42
Earthmine	Garbage Bin	122	100
Earthmine	Traffic Light	175	75
Earthmine	Parking Sign	134	111
Earthmine	Traffic Cone	52	22
AAM	Car	729	557
AAM	Traffic Light	1172	872
AAM	Triangular Road Sign	163	161
AAM	Truck / Van	188	179
AAM	Road Light	1461	1350
AAM	Telegraph Pole	937	899
AAM	Rectangular Road Sign	668	613

Table 5.1: Object Classes used for Object Detection Evaluation

Four different experiments are undertaken for each object class. Table 5.2 details each of the experimental configurations. The “Original” experiment generates baseline results for the CSHF without depth extensions. These baseline results are compared against the results from the three other experiments using the respective depth extensions to evaluate if object detection recall and / or precision are improved.

In the “Feature Type” column of table 5.2, *GL* refers to the original patch descriptor extraction method as used by Gall *et al.* (2009) and described in section 5.2.2. As in the original CSHF evaluation, 16×16 pixel patch descriptors are generated. Where patch scaling is used, the features from the extracted region are resized to the required pixel dimensions. *DD₅FX* refers to the depth based features described in section 5.2.2 and the depth difference calculation of equation 5.17 sets $\alpha = 1$ irrespective of the object type.

The “Scaled–DW” and “Scaled+DW” experiments use the patch / offline scaling extension described in section 5.2.1 without and with depth weighting (described in section 5.2.3) respectively. Patch / offset scaling is evaluated with and without vote depth weighting to determine if depth weighting actually does result in the intended improvement in detection

precision.

The “Scaled DD_5FX ” experiment uses the depth based features described in section 5.2.2 *instead of* the colour based features used in the original CSHF evaluation (Gall *et al.*, 2009), and combined with patch / offset scaling. In all three of the scaled patch / offset configurations, patch dimensions of 0.5×0.5 metres are used.

Experiment	Patch / Offset Scaling	Depth Weighting	Feature Type
Original	No	No	GL
Scaled-DW	Yes	No	GL
Scaled+DW	Yes	Yes	GL
Scaled DD_5FX	Yes	No	DD_5FX

Table 5.2: The four experimental configurations for object detection on each object type.

Each experiment for each object category follows a standard methodology. Positive patch descriptors are generated from fifty random locations within each example’s bounding box; the random location of a patch determining its centre. Instead of using patches from every pixel position in each example, a fixed number of patches are extracted from each example for two reasons. Primarily, to avoid positively biasing larger examples, but secondarily, due to memory constraints. The square dimensions of a patch means that up to three quarters of its area can reside outside of an object’s bounding box (if the patch’s centre is in the corner of an object’s bounding box). The ground-truth bounding rectangles closely fit the actual boundaries of their objects. Extracting patch descriptors just inside the rectangular ground-truth boundaries means that the actual boundary of the examples, and thus the shape of the examples, is encoded. This is important as the bounding shape of an object is a useful characteristic in helping to detect the presence of objects (Ballard, 1981). This feature of patch extraction means that ground-truth examples are given in their local image surrounds so that sufficient image background can be encoded by the patch descriptors. In previous work, such context has been found to enhance detection accuracy due to the frequency with which certain object types are found in particular locations (*e.g.* cars on roads) (Wang *et al.*, 2012; Kim and Medioni, 2011; Divvala *et al.*, 2009; Torralba and Sinha, 2001).

The same number of negative patch descriptors are generated from the *same images* containing the positive examples, but sampled from random locations such that none of the negative patches overlap with any of the positive example ground-truth bounding boxes. The pixel dimensions of each patch extracted from the negative data are found in accordance with equation 5.12. This requires that the negative data have coregistered depth data. Although possible to simply extract fixed pixel dimension patches from random locations in the negative images, the characteristics of these patches would then be independent of depth. Since the character of the positive training patches is not independent of depth (due to the patch scaling

process), it is important that no systematic difference be introduced between the positive and the negative patches that might be represented in the extracted patch descriptors and then learned by the CSHF. As with the positive patches, the negative patches are scaled prior to feature extraction. Part offsets are only generated for positive part patches.

A pseudo random number generator is used to generate the locations for the fifty patches extracted from each example. This is initialised using the same seed across all experiments so that the variation in results is explainable only in terms of the different experimental configurations. All patch descriptors and part offsets are generated in accordance with the current experimental parameters.

Five-fold cross validation is used to evaluate detection accuracy. This is a good compromise between the CSHF in each round drawing upon a large enough dataset to represent the object type, and the length of time required for training. The cross validation rounds partition the images containing the ground-truth examples into a set of training images, and a set of validation (or query) images. For training the CSHF in each round, the positive and negative patch extract sets are selected from the training images. The sizes of the positive and negative patch extract sets are always equal, but the specific number of extracts used to train the CSHF varies for each round because of the different number of objects in each image.

Each query image in the validation set is processed by generating patch descriptors at every pixel location having coregistered depth values < 100 metres. Locations with no coregistered depth (*e.g.* sky) or at greater distances are ignored. Patch descriptors are generated using the appropriate feature extraction parameters for the experiment. The patches are parsed by the CSHF to determine vote locations. If required for the experiment, the votes are weighted according to equation 5.20 before being aggregated into the query image’s vote map. The vote map \mathcal{H} produced from each query image is then Gaussian smoothed with a 3×3 pixel kernel ($\sigma^2 = 9$) which helps to more accurately obtain the maxima in the Hough map since neighbouring pixel votes are discretised. An alternative method (not evaluated here) would have been to cast votes into smaller scale Hough maps before scaling up. It was noted that at larger scales, inaccuracies in voting locations are also larger and that commensurately larger Gaussian kernels should be used to account for these uncertainties. However this aspect of vote aggregation was not tested here due, in part, to the added computational burden of convolving images with dynamically scaling kernels.

In the original CSHF evaluation, Gall *et al.* (2009) carried out detection over n different scales producing n different Hough vote maps. In these experiments, only single response maps are produced for each query image so that the effect of the depth extensions can be clearly demonstrated.

5.3.1 Estimating Detected Object Bounds

The relative confidence of object detections for each query image \mathcal{Q} is shown in its corresponding vote map \mathcal{H} . The points in \mathcal{Q} specifying hypothesised object detections are given by the corresponding maxima in \mathcal{H} . These points are found by successively parsing \mathcal{H} to find the current maximum $\mathbf{x} = \langle x_0, x_1 \rangle$, finding the corresponding depth at \mathbf{x} , and then calculating the scaled average bounding box dimensions of an object of the type positioned such that its centre base is at \mathbf{x} . The area within the bounding box in \mathcal{H} is suppressed so that subsequent maxima are found outside of these bounds. This ensures that only the largest mode within the local region of each potentially detected object is used to estimate its location.

The detection bounding box $\mathbf{d} = \langle x, y, h, v \rangle$ at a mode \mathbf{x} is defined as

$$d_h = \lambda \frac{m_h}{\text{depth}(\mathcal{Q}, \mathbf{x})} \quad (5.21)$$

$$d_v = \lambda \frac{m_v}{\text{depth}(\mathcal{Q}, \mathbf{x})} \quad (5.22)$$

$$d_x = x_0 - \frac{d_h}{2} \quad (5.23)$$

$$d_y = x_1 - d_v \quad (5.24)$$

assuming a top left pixel origin, where $\mathbf{m} = \langle m_h, m_v \rangle$ gives the average horizontal and vertical dimensions of the object type in the same units as used in the range imagery (nominally metres). This is either specified manually, or found upon processing the object type dataset.

Twenty candidate detection boxes are identified for each vote map setting a maximum number of allowed detections per query image and increasing the likelihood of locating most of the ground-truth objects in each query image. Each detection box is associated with a confidence score given as the value of \mathcal{H} at \mathbf{x} giving a set of detections $\mathcal{D}_{\mathcal{Q}} = \{(\mathbf{d}, \mathcal{H}_{\mathbf{x}})\}$ for each query image \mathcal{Q} . Overlapping detection areas in $\mathcal{D}_{\mathcal{Q}}$ are removed by retaining the highest confidence detections. Where a lower confidence detection box \mathbf{t} has at least half of its area overlapped by a higher confidence detection box \mathbf{d} , \mathbf{t} is removed from the list of candidate detections. In cases where \mathbf{t} overlaps \mathbf{d} with less than half of its area, bounding box \mathbf{t} is modified in either its vertical or horizontal dimension (but not both) so that it no longer overlaps \mathbf{d} .

Because each view contains at least one instance of the object type being used for evaluation, the confidence values for all detections in a view are standardised to fall into the range $[0, 1]$. This results in a modified detection set $\mathcal{D}'_{\mathcal{Q}}$ where $\forall \mathbf{d}, \mathbf{t} \in \mathcal{D}'_{\mathcal{Q}} \mid \mathbf{d} \cap \mathbf{t} = \emptyset$.

The depth data are exploited to assist in the estimation of detected object bounding boxes for all of the experiments, including the original baseline experiment. This is justified because the method does not affect the prior determination of votes for the potential object reference points, and bounding box estimation is used only to generate object location hypotheses. The accuracy of bounding box estimation is controlled across the experiments because the use of the depth data to determine the object bounds is affected only by the placement of the Hough space maxima. Alternative methods for determining the bounds of detected objects were considered, including methods that use the vote support (*i.e.* the locations of the patches that vote for the identified maxima) to determine the object bounds. However, such methods would have made it harder to appraise the extent to which the existing depth extensions dictate object detection and localisation accuracy.

After the completion of cross validation, the combined ground truth and detection boxes for the object dataset are processed according to the object detection evaluation criteria described in section 3.5.2. One hundred threshold values taken evenly from the range $[0, 1]$ are used to generate 100 recall and precision data points, which are then used to plot precision-recall graphs. The upper limit of 20 permissible object detection hypotheses per view means that the calculated recall and precision values do not in general range from zero to one. The relative placement of the precision-recall curves from the different experiments indicates the relative detection accuracy achieved by the different experimental configurations for the object type.

5.3.2 Earthmine Depth Image Preprocessing

The quality of the depth data in the AAM dataset are generally good due to the use of high precision laser scanners. The depth data in the Earthmine dataset (see section 3.2) were generated using stereophotogrammetry which has lower precision. As found in section 4.3, the Earthmine depth data are noisy and much of the depth information for individual objects (especially small or slim objects) is missing. In order to use the Earthmine depth data to evaluate the depth extensions the depth images are preprocessed so as to reconstruct portions of the missing depth data.

Figures 5-3(a) and 5-3(b) show a single view from one of the Earthmine panoramas giving the colour information and the generated coregistered (stereoscopic) depth information respectively (lighter regions in the depth image are closer to the camera plane). The image in figure 5-3(b) shows that depth information is missing or erroneous in several regions, especially for the traffic lights where the depth information for the light poles is completely

missing, and the traffic light cowlings are not accurately depicted. Larger surfaces such as the ground plane and the building façades are more accurately represented.

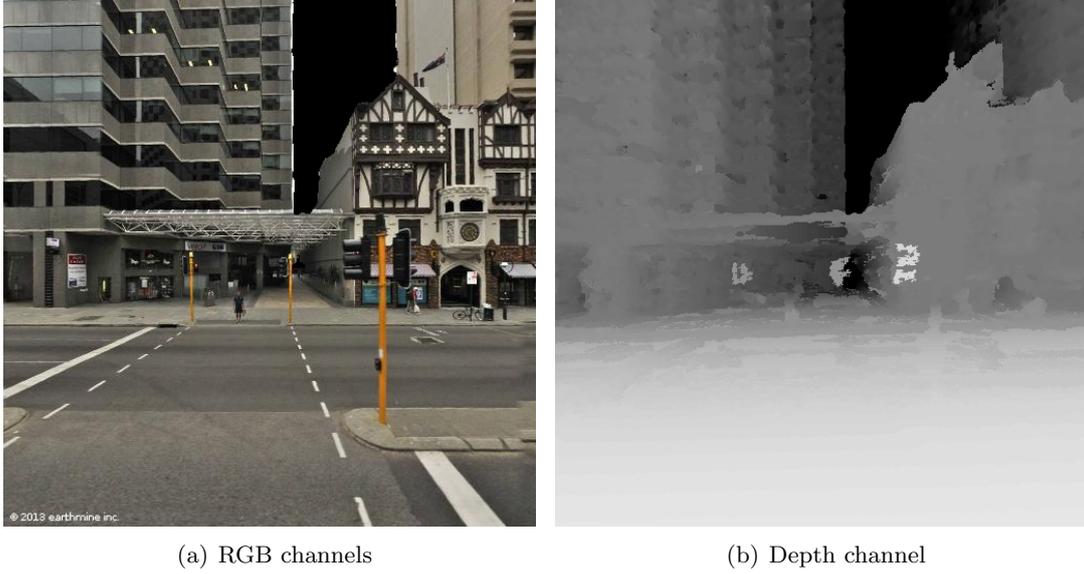


Figure 5-3: Missing and erroneous depth information in the Earthmine dataset.

All of the object types in this evaluation sit on the ground plane. In the Earthmine data, while individual objects are often poorly represented in depth, the ground plane itself is not. The depth based extensions require the objects of interest to be represented in depth as completely as possible – especially over the projected surface of the object, to help discriminate the object from its more distant background. The Earthmine depth information is incomplete for some objects, hence these missing depth data must be reconstructed. While it is not possible to fully and accurately reconstruct the depth silhouette of these objects (since this would require these objects to already be detected and localised), a coarse estimate can help to prevent parts of the object from being ignored at detection time when employing the depth extensions.

The detection framework is class specific and the preprocessing requires the estimated average real world dimensions of the object class $\hat{\mathbf{m}}_c$ to be modelled by the CSHF. At each pixel \mathbf{x} in a depth image \mathcal{I} , the scaled size of the object $\hat{\mathbf{s}}_c$ is calculated as

$$\hat{\mathbf{s}}_c = \frac{\lambda}{\text{depth}(\mathcal{I}, \mathbf{x})} \hat{\mathbf{m}}_c \quad (5.25)$$

with λ being the focal length of the camera measured in pixels.

The depth at \mathbf{x} is copied horizontally left and right and vertically down the image towards the ground plane by the pixel dimensions $\hat{\mathbf{s}}_c$, or until nearer depth values are encountered.

The depth values are only copied downwards in the image because the object types being tested all sit on the ground plane and the camera up vector points upwards in the image. Growing the depth regions upwards would generate large spurious regions of invalid depth values. Most ground-truth objects have some small number of “seed” pixels giving the depth of the object at its top.

When applied to the depth image shown in figure 5-3(b) using the mean dimensions of the traffic light object type (as estimated from the ground-truth), the image shown in figure 5-4 is generated.

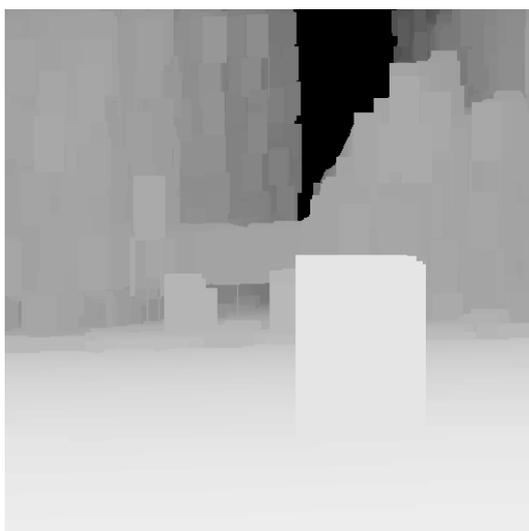


Figure 5-4: After preprocessing the depth image in figure 5-3(b) using the average dimensions of the Earthmine “Traffic Light” object type.

In the unprocessed Earthmine depth maps, a seed pixel for an object may be located at either of the object’s horizontal extents. The corresponding depth value must therefore be copied by a full scaled object width both to the left and to the right in order to ensure that the newly expanded region in depth intersects where the object could be present in the image. As a result, the processed depth images have much more expansive areas of similar depth than would be present if the depth data were not incomplete. This is the reason for the large foreground block in figure 5-4; it is generated from the depth information from the foreground traffic light.

Given the nature of the depth extensions, over compensating for the missing depth information is preferable if the alternative is that depth information remains incomplete; the detection accuracy can only be improved using the depth based extensions if sufficient depth information about the objects is available.

5.4 Results and Analysis

Each of the following sections presents results for the experiments evaluated against one of the twelve object categories listed in table 5.1. Each section shows the generated precision-recall graph for the four different experimental configurations. This is followed by an example query image that is indicative of the dataset showing the ground-truth objects delimited with blue bounding boxes. The resulting contrast scaled Hough vote maps are then shown for that query image. These show the qualitative differences between the voting patterns generated for the hypothesised object detections using the different depth extensions compared with the original unchanged CSHF detector of Gall *et al.* (2009).

Each section includes an analysis of the results in terms of the ability of the depth extensions to enhance detection accuracy given the characteristics of the particular object type and the dataset in general (AAM or Earthmine). An overall summary of the results for all of the object types follows in section 5.4.13.

5.4.1 Earthmine “Car”

Figure 5-5 shows precision-recall graphs for the results from the four experiments listed in table 5.2 carried out on the Earthmine “Car” object type.

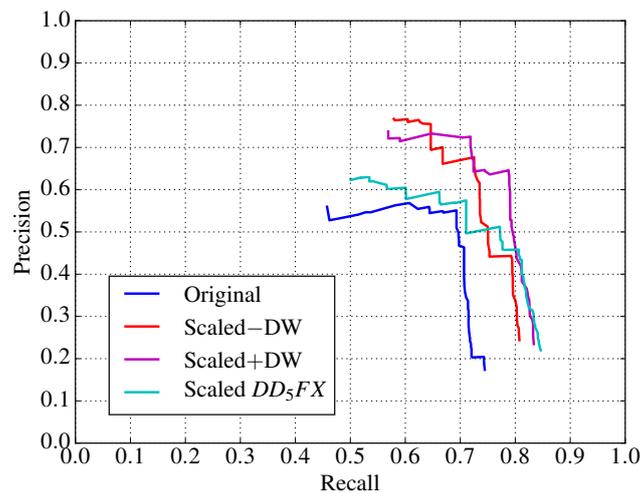


Figure 5-5: Precision versus Recall for Earthmine “Car” detection using CSHF with and without depth extensions.

Figure 5-5 shows that for this object type, all of the depth extensions improve detection recall, and that precision is also higher at every recall level. That is, the depth extensions allow for more of the ground-truth objects to be detected than in the baseline case and fewer false positive errors are made in doing so.

When combined with the patch / offset scaling extension, the original 8192 dimension feature vectors generated from the colour images give higher accuracy than the ten dimensional feature vectors generated from the depth images. Even though the depth data for the Earthmine dataset are poor quality, the larger average size of the objects in comparison to other scene elements means that the depth based feature improves good detection accuracy for this object type. The larger size of the objects also allows the depth weighting extension to further improve detection accuracy because patch descriptors are generated from a larger subregion having the same depth as the reference point the parts are voting for.

An example query image from the Earthmine “Car” dataset showing ground-truth object location(s) with blue rectangles is shown in figure 5-6(a). Figures 5-6(c) through 5-6(f) show contrast scaled Hough vote maps from the four different experimental configurations in table 5.2.

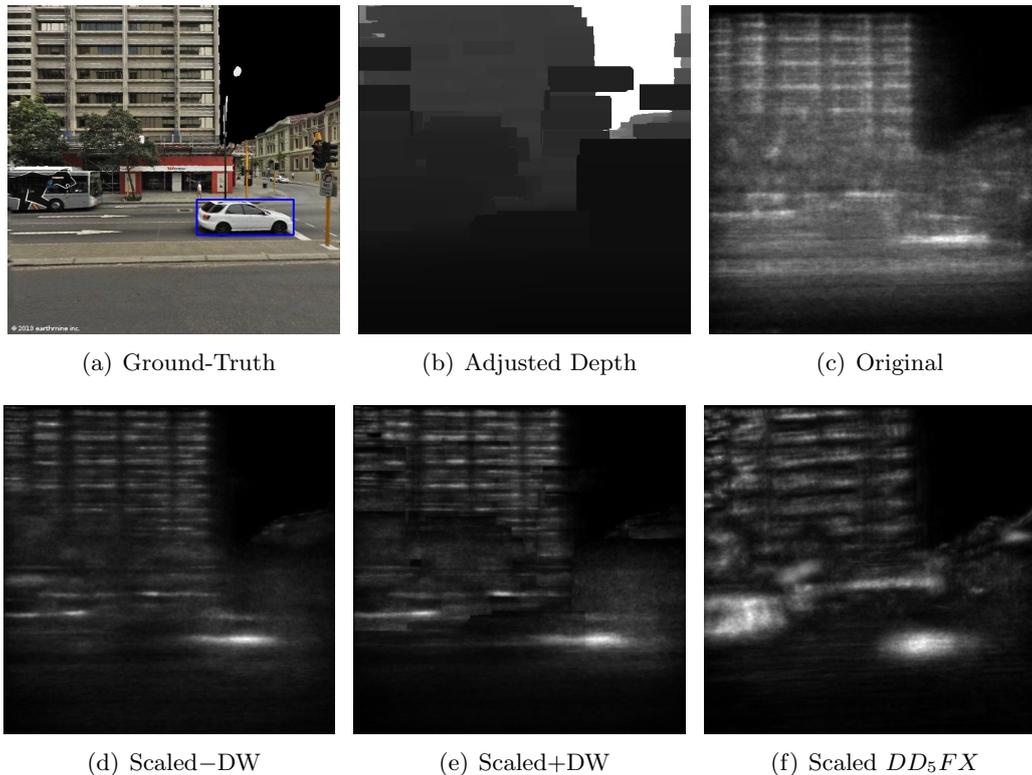


Figure 5-6: A representative query image from the Earthmine “Car” dataset, and generated Hough vote maps for the four different experimental configurations.

Figures 5-6(d) and 5-6(e) show much more “compact” voting regions than in the baseline results shown in figure 5-6(c). The much brighter regions around the kerb edges and over the building façade in the baseline results of figure 5-6(c) compared to the patch / offset scaling results in figure 5-6(d) show that the many non-scaled patch descriptors from these regions are erroneously being classified as possible car parts in the baseline case. Patch scaling is acting to reduce the number of false positive patch descriptor classifications.

As hypothesised, the effect of depth weighting seen in figure 5-6(e), is to reduce the magnitude of the votes being cast in regions where the difference in depth between the extracted parts and the voting locations are too large. This is indicated by the darker regions in this figure when compared with figure 5-6(d). The higher contrast edges to these darker regions in figure 5-6(e) are due to the results of the processing of the Earthmine depth images prior to cross validation.

Figure 5-6(f) shows that the depth based feature produces vote map results that are not as washed out as in the original case as seen in figure 5-6(c), but the vote clusters are not as compact. This is likely due to the much lower dimensionality of the depth based features and the fact that colour information is being ignored.

5.4.2 Earthmine “Garbage Bin”

Figure 5-7 shows precision-recall graphs for the results from the Earthmine “Garbage Bin” object type. When using the original colour based feature descriptors, the patch / offset scaling extension significantly improves detection accuracy. As the detection confidence levels increase, recall accuracy remains high in the case of the patch / offset scaling results, while recall accuracy drops very quickly in the original configuration.

For the results in the case of the two patch / offset scaling configurations using the original colour based features, contrary to expectations, the depth weighting extension does not increase precision at a given level of recall. In fact, using the depth weighting extension for this object type results in a reduction in precision. This could be due to the smaller size of the objects compared to the results observed for the Earthmine “Car” object type.

The images in figure 5-8 show results for an example query image from this dataset. Figure 5-8(c) shows the vote map generated from the original configuration. The brightest region in this image is at the base of the pole at the left of the image. This indicates that the garbage bin parts are being detected at the wrong scale. The true location of the ground-truth garbage

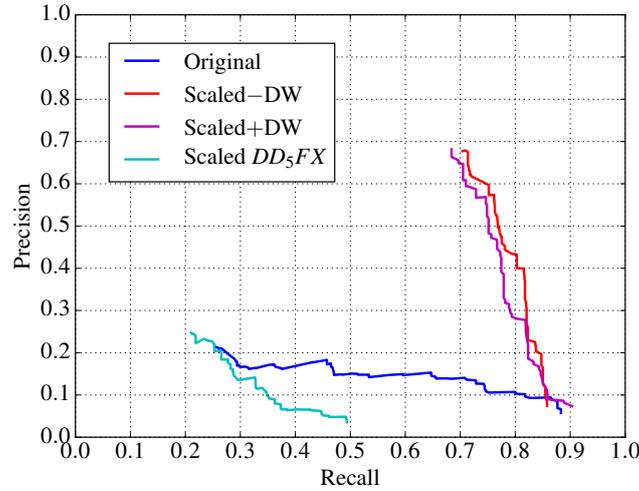


Figure 5-7: Precision versus Recall for Earthmine “Garbage Bin” detection using CSHF with and without depth extensions.

bin is darker and less well defined. Figure 5-8(d) shows the results using patch / offset scaling (without depth weighting). In this vote map, the true location of the ground-truth garbage bin is clearly found, and the base of the pole to the left of the image is no longer being erroneously detected. Patch / offset scaling is having the desired effect in this case as it more accurately detects the locations of the garbage bins at the scale they appear in the query image.

When the depth weighing extension is introduced, the effect shown in figure 5-8(e) is to introduce many more bright regions indicating possible detection hypotheses at a higher confidence level. This is due to the contrast normalisation on the images and this actually indicates that the true location of the ground-truth garbage bin received votes of a lower aggregate magnitude. This implies that the relatively higher vote response seen in figure 5-8(d) is because it was generated from more patch descriptors extracted from locations dissimilar in depth to the voting location. This further implies that the contextual surrounds of the object are having a greater contribution to the object’s detection.

The vote map for this query image resulting from the use of the depth based features shown in figure 5-8(f) shows that the depth based features are not well suited to this dataset. A large number of poorly defined vote maxima are generated, and the true location of the object is not confidently determined.

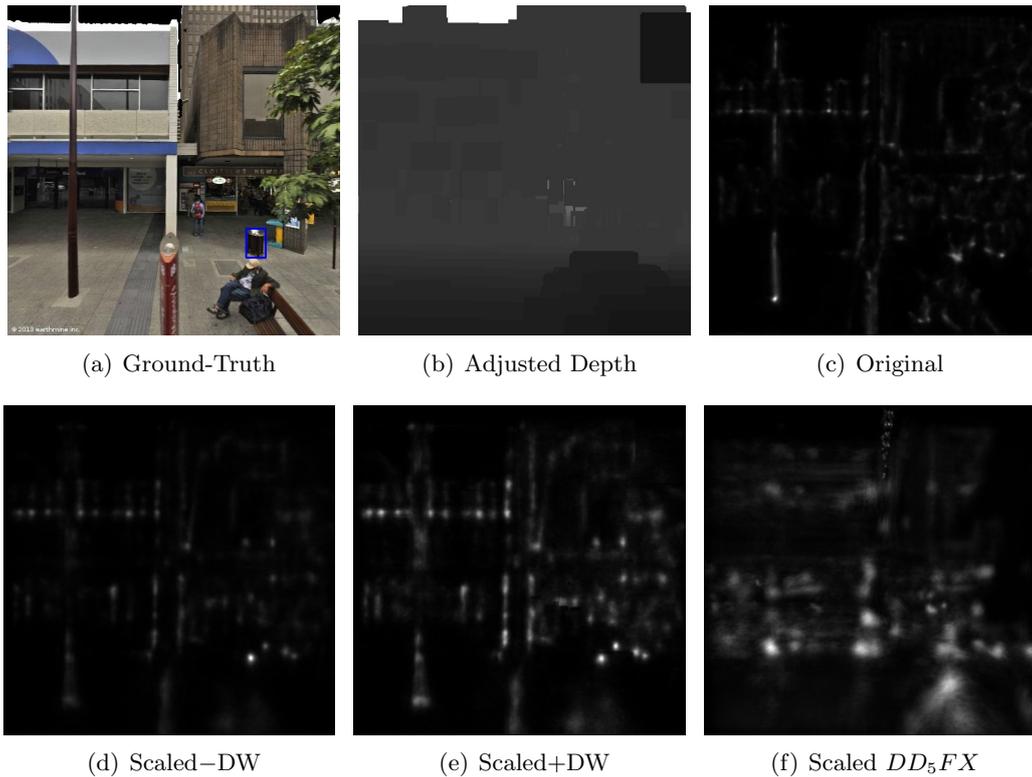


Figure 5-8: A representative query image from the Earthmine “Garbage Bin” dataset, and generated Hough vote maps for the four different experimental configurations.

5.4.3 Earthmine “Traffic Light”

Figure 5-9 shows precision-recall graphs for the results from the Earthmine “Traffic Light” object type. For this object type, the patch / offset scaling extension results in slightly worse detection accuracy results compared to the original method. Again, depth weighting decreases rather than increases precision. The depth based features are far worse than the original colour based features.

The images showing the generated vote maps for the different experiments on an example query image are shown in figure 5-10. These images reflect the decrease in detection accuracy evidenced by the precision-recall graphs in figure 5-9. In all three cases where the depth extensions are applied, the response maps are more washed out indicating lower confidence detections than in the example vote map generated in the original case seen in figure 5-10(c).

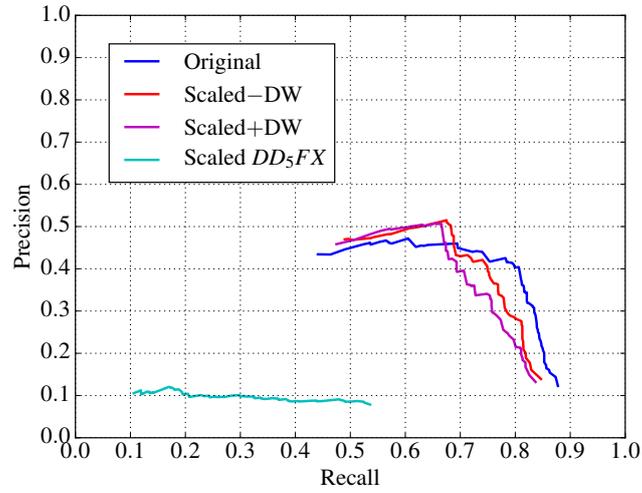


Figure 5-9: Precision versus Recall for Earthmine “Traffic Light” detection using CSHF with and without depth extensions.

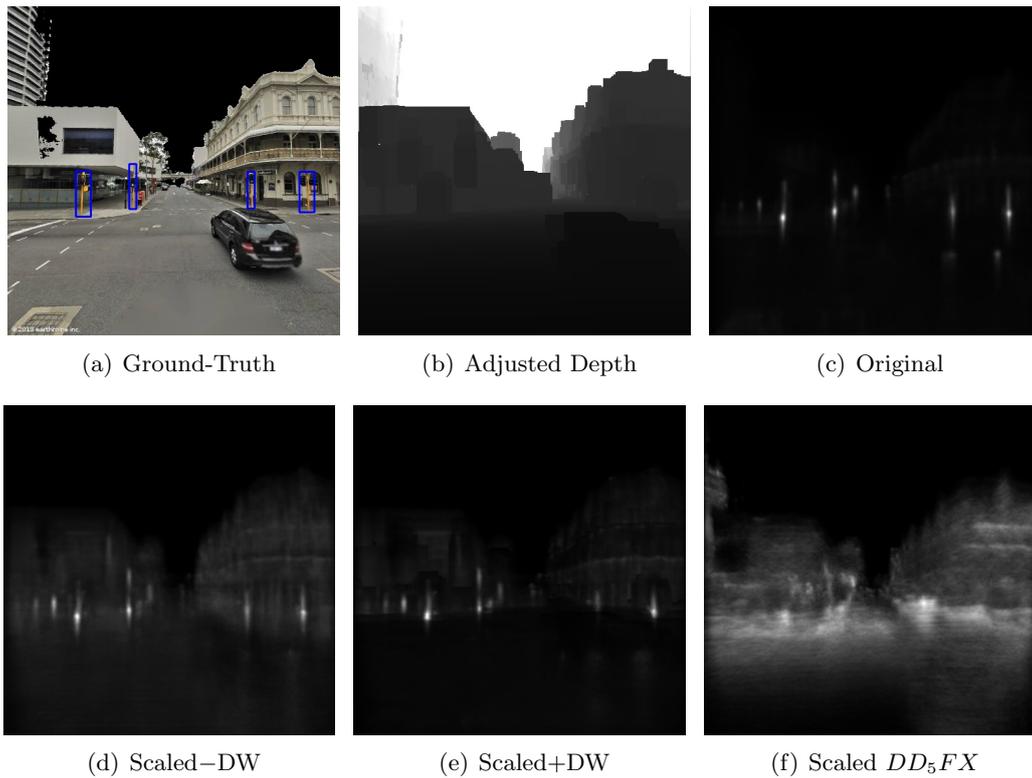


Figure 5-10: A representative query image from the Earthmine “Traffic Light” dataset, and generated Hough vote maps for the four different experimental configurations.

5.4.4 Earthmine “Parking Sign”

Figure 5-11 shows precision-recall graphs for the results generated from the Earthmine “Parking Sign” object type. Patch / offset scaling improves precision accuracy when compared to

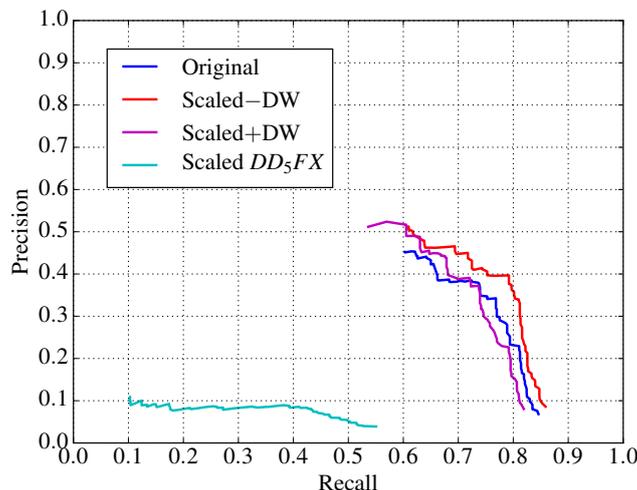


Figure 5-11: Precision versus Recall for Earthmine “Parking Sign” detection using CSHF with and without depth extensions.

the baseline configuration but only when the depth weighting extension is disabled. Recall accuracy is not significantly improved. The depth based features for this object type again fail to offer discriminative power comparable to the original colour based features. The continued inability of the depth based features to offer detection accuracy comparable to the colour based features is almost certainly due to the depth image preprocessing carried out on the Earthmine images.

Example vote maps generated for an example query image are shown in figure 5-12. The vote map generated for the patch / offset scaling experiment (without depth weighting) shown in figure 5-12(d) shows that there are fewer false positive detections than in the case of the vote map generated from the original CSHF displayed in figure 5-12(c). For this example query image, the effect of depth weighting shown in figure 5-12(e) is qualitatively to reduce the aggregated magnitude of the false positive detections (when compared to the results in figure 5-12(d)) — which should increase detection precision. However, this single example’s results are contrary to the overall results that show that depth weighting acts to decrease the relative accuracy in precision overall.

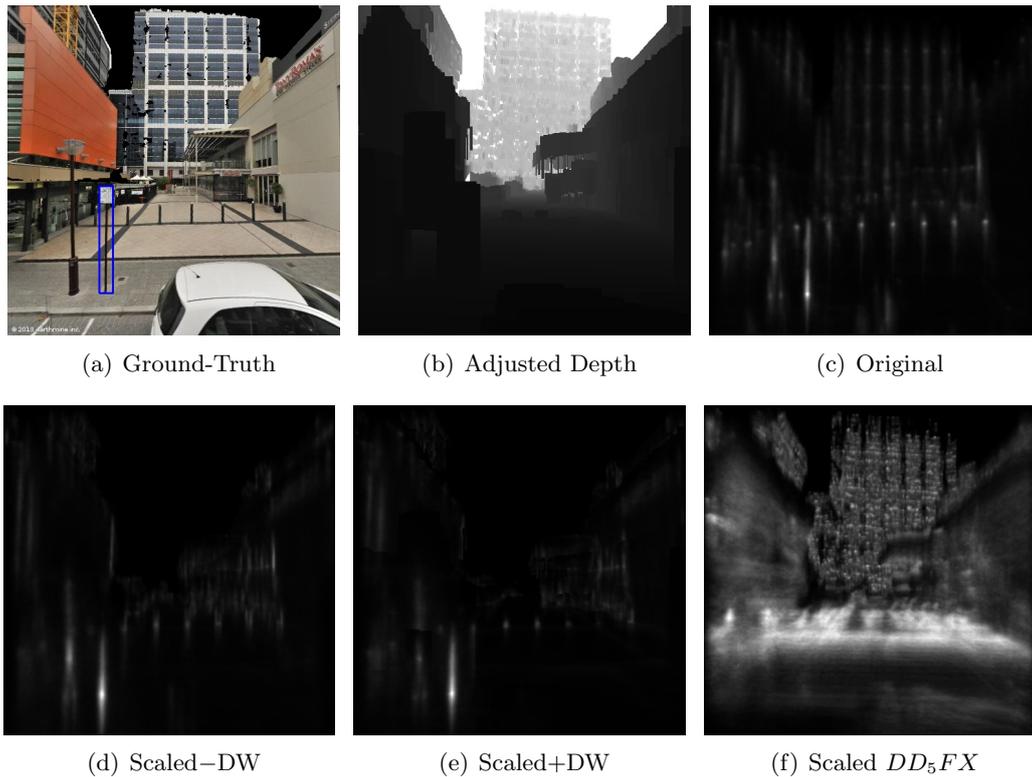


Figure 5-12: A representative query image from the Earthmine “Parking Sign” dataset, and generated Hough vote maps for the four different experimental configurations.

5.4.5 Earthmine “Traffic Cone”

Figure 5-13 shows precision-recall graphs for the results from the experiments conducted using the Earthmine “Traffic Cone” object type. The precision-recall curves show that the patch / offset scaling extension improves both recall and precision at all confidence thresholds over the original non-depth extended version of the CSHF. However, depth weighting only improves precision beyond this at the higher confidence thresholds (where recall is lower). Again, the depth based features offer very poor detection accuracy compared to the original colour based features.

The images in figure 5-14 show generated response maps for each of the experiments for an example query image. In all four of the vote maps, the two ground-truth traffic cones are detected with the highest confidence detections. Precision is qualitatively highest in figure 5-14(e) where much more of the response map is darker and the few maxima are brighter and more tightly clustered.

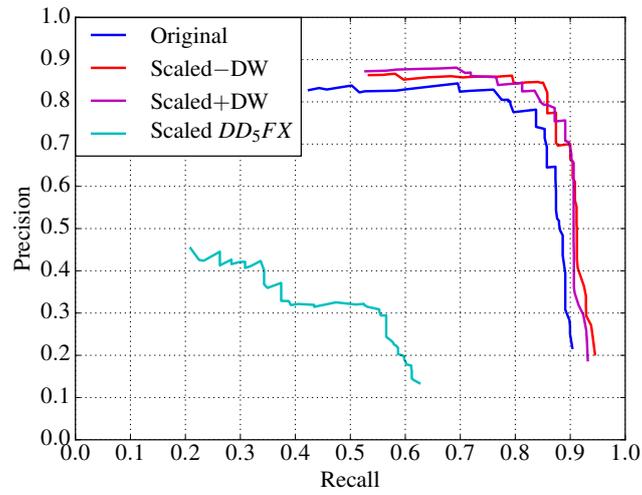


Figure 5-13: Precision versus Recall for Earthmine “Traffic Cone” detection using CSHF with and without depth extensions.

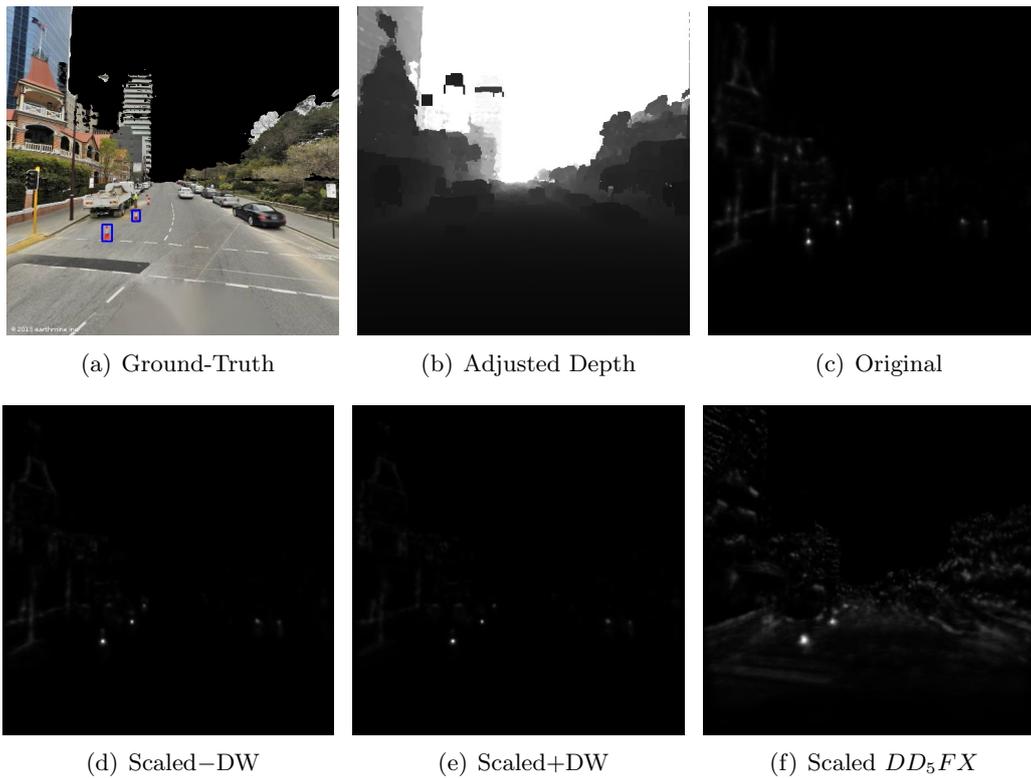


Figure 5-14: A representative query image from the Earthmine “Traffic Cone” dataset, and generated Hough vote maps for the four different experimental configurations.

5.4.6 AAM “Car”

Precision-recall graphs are shown in figure 5-15 for the results of the experiments carried out on the AAM “Car” object type.

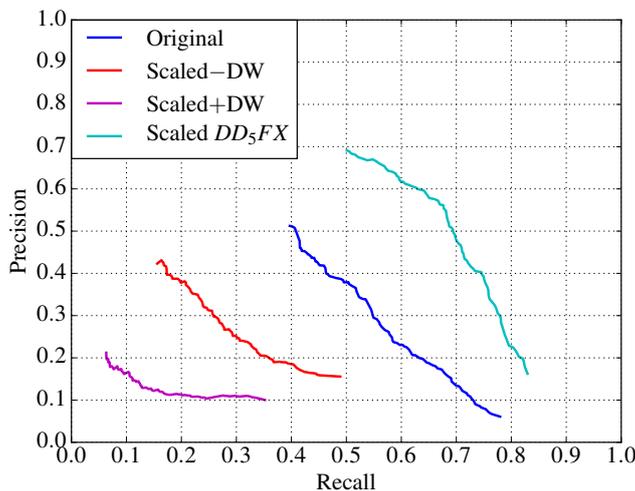


Figure 5-15: Precision versus Recall for AAM “Car” detection using CSHF with and without depth extensions.

For this dataset, both recall and precision detection accuracy is much higher using the depth based features in lieu of the original colour based features even though the depth based features are encoded using only ten dimensions as opposed to the 8192 dimensional colour based features. This is likely due to the fact that the AAM depth data are much more accurate than the Earthmine data and depth image preprocessing is not necessary for the AAM depth data. Given this improved accuracy in depth, it is unexpected that the patch / offset scaling extension results in levels of detection accuracy that are much lower than the baseline configuration. The higher accuracy depth data should lead to more precise scaling of object part patches and their associated offsets resulting in tighter clusters of aggregated votes having larger overall values.

The results for the depth weighting extension are also counter-intuitive; depth weighting the patch / offset scaling votes does not improve precision accuracy. Again, this indicates that significant votes are being generated from query image subregions that are not located within the extents of the projected surface of the objects.

The generated vote maps in figure 5-16 provide some insights into why the patch / offset scaling extension results in much lower detection accuracy. In the original response map

shown in figure 5-16(c), the three bright areas denote the ground-truth locations of the cars in the query image. Corresponding bright regions are also present in the response map generated in the patch / offset scaling experiment without depth weighting shown in figure 5-16(d), but the relative brightness of these locations is lower.

Given the very noisy colour images in the AAM datasets, it may be the case that patch scaling decreases rather than increases the discriminative ability of the CSHF. By maintaining fixed pixel dimensions in the original scheme, the scale of the patches may itself be contributing to increased detection accuracy if most of the examples in the dataset are present within a very narrow range of scales. This is evidenced from the histogram of object sizes for this dataset presented in figure 4-29 on page 115. Further support for this explanation is given by the fact that when using the depth based features instead of the colour based features, detection accuracy is improved.

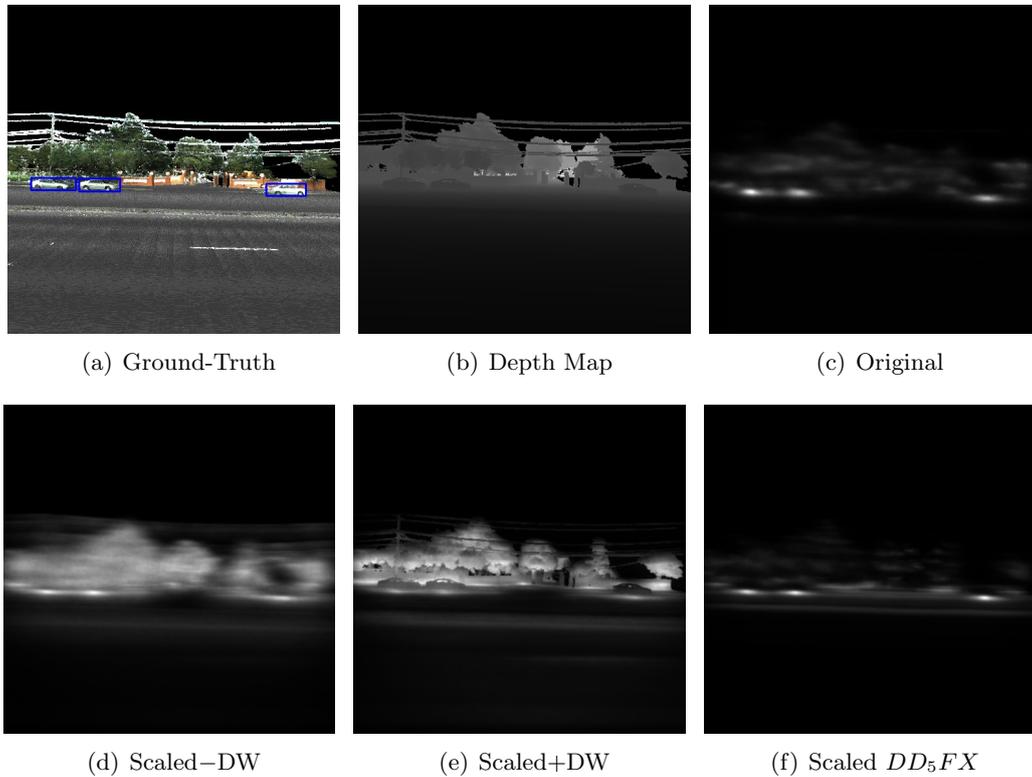


Figure 5-16: A representative query image from the AAM “Car” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.

5.4.7 AAM “Traffic Light”

Precision-recall graphs showing the results for the AAM “Traffic Light” object type are shown in figure 5-17.

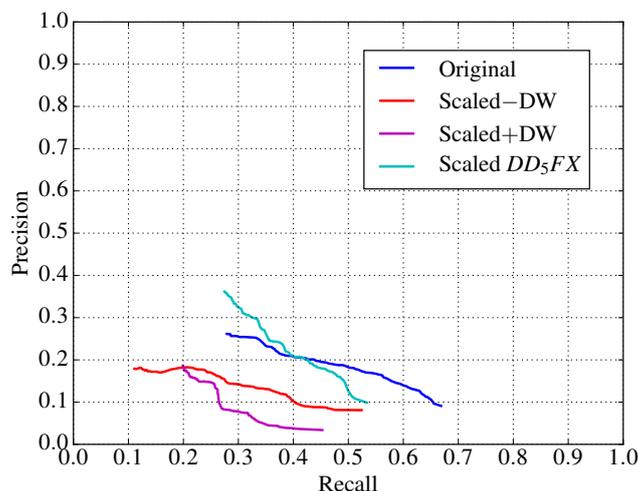


Figure 5-17: Precision versus Recall for AAM “Traffic Light” detection using CSHF with and without depth extensions.

For this object type, none of the depth extensions provided detection accuracy improvements beyond the baseline results given by the original CSHF configuration. Overall, detection accuracy was poor in all of the CSHF configurations. Again, given the poor detection performance by the patch / offset scaling extensions, it is likely that the fixed dimension patches used in the original CSHF configuration encode salient information about the object type at its most commonly occurring scale in the imagery. The size distribution data for the ground-truth examples shown in figure 4-33 on page 118 bear this out; most of the examples fall within a relatively narrow range of pixel dimensions.

As in the results for the AAM “Car” object type, the better detection accuracy provided by the depth based features (compared to the colour image based features) offers further corroboration of this explanation. Qualitative results from an example query image for this dataset are shown in figure 5-18.

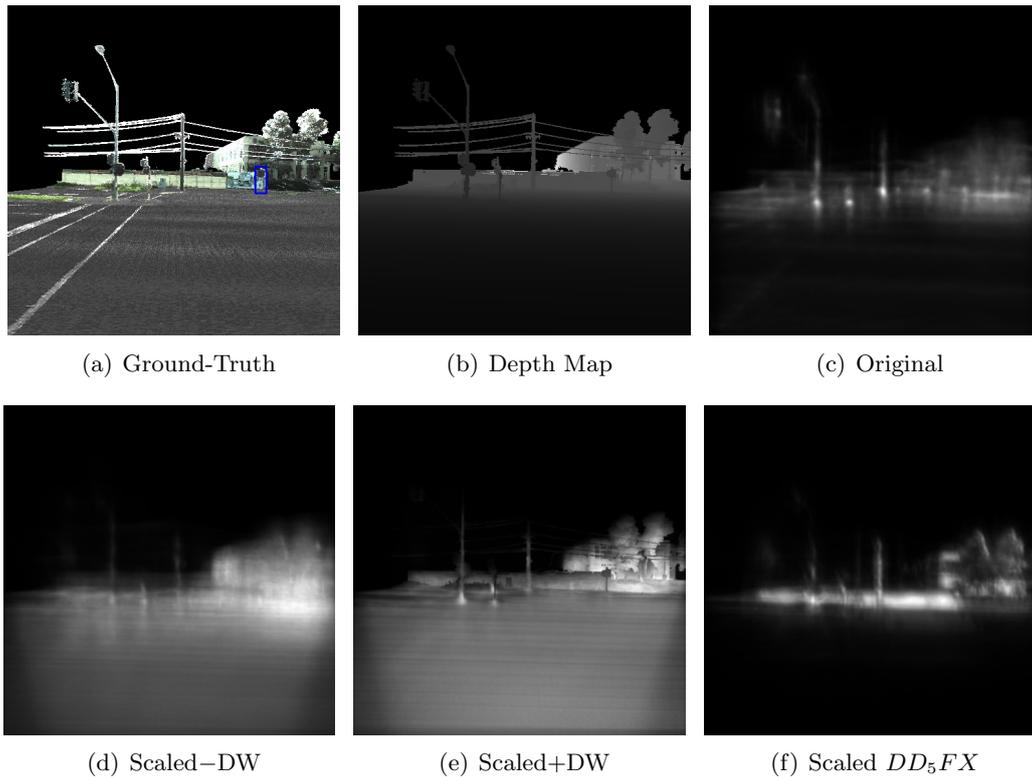


Figure 5-18: A representative query image from the AAM “Traffic Light” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.

5.4.8 AAM “Triangular Road Sign”

Figure 5-19 shows precision-recall graphs for the results from the AAM “Triangular Road Sign” object class. Similar object detection accuracy is seen in the results for the AAM “Triangular Road Sign” object type as was observed in the cases of the AAM “Car” and “Traffic Light” object types. None of the depth extensions are able to improve detection accuracy over the baseline configuration, and again, the size distribution of the ground-truth examples (as shown in figure 4-37 on page 121) falls within a very narrow range. Again, the depth based features give detection accuracy that is better than the patch / offset scaled colour image based features, but this is most likely due to the much higher quality depth information compared to the colour information. The images in figure 5-20 show the Hough vote maps generated from the different experimental configurations for an example query image.

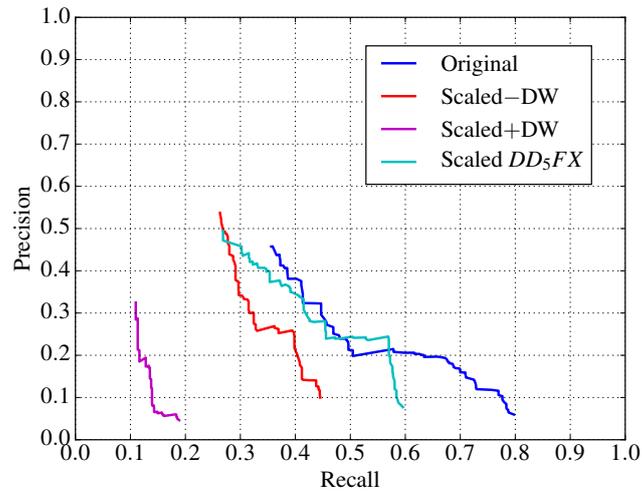


Figure 5-19: Precision versus Recall for AAM “Triangular Road Sign” detection using CSHF with and without depth extensions.

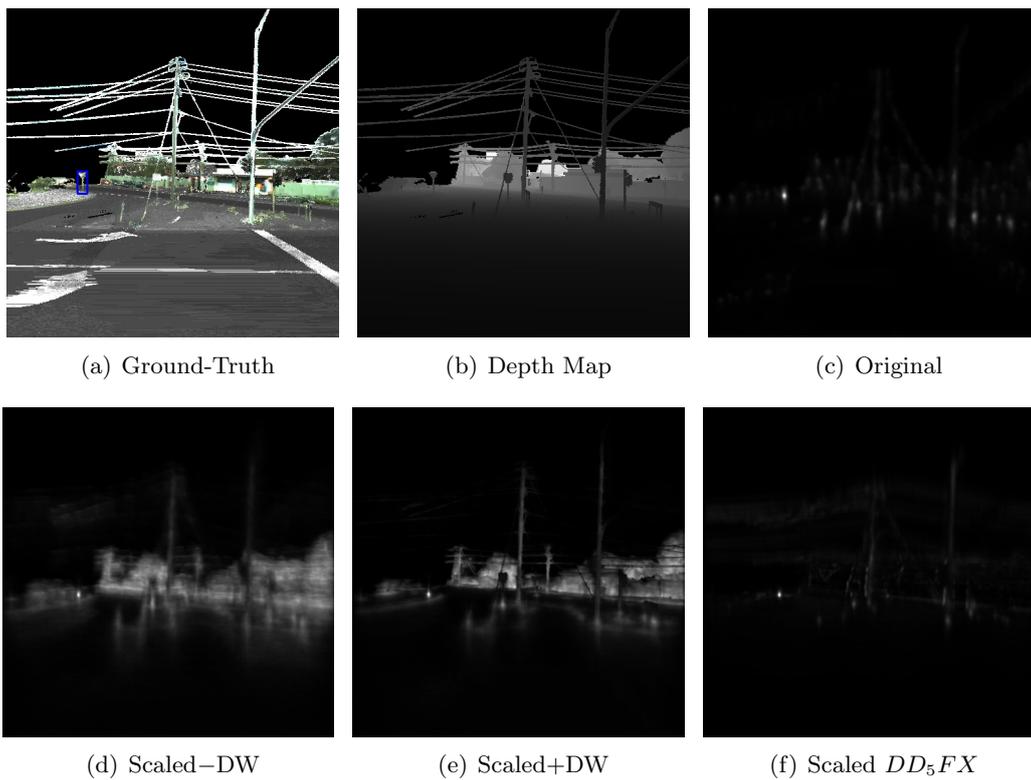


Figure 5-20: A representative query image from the AAM “Triangular Road Sign” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.

5.4.9 AAM “Truck / Van”

Figure 5-21 shows precision-recall graphs for the results from the experiments carried out on the AAM “Truck / Van” object type.

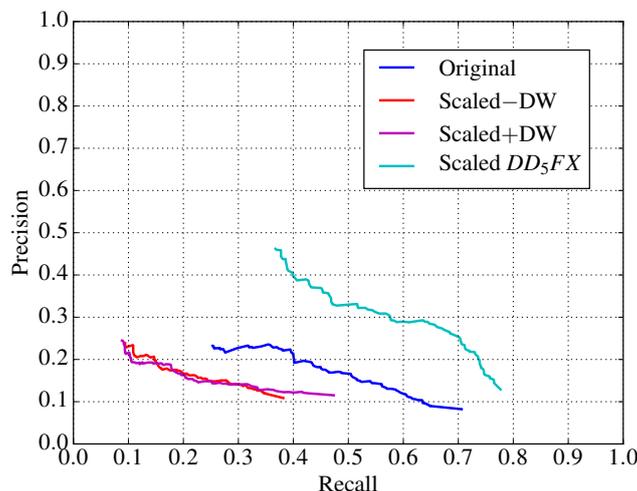


Figure 5-21: Precision versus Recall for AAM “Truck / Van” detection using CSHF with and without depth extensions.

The results for this object type differ from the results seen thus far for the other AAM object types. The detection accuracy afforded by the patch / offset scaled colour image based patch descriptors (with or without depth weighting) is still worse than in the case of the non-scaling CSHF experiment. However, the depth based features this for this object type provide significantly improved detection accuracy over the baseline results. There are two main differences explaining this result. Firstly, the nature of the object type is more complex when represented in depth — the shape of the object type is arguably more distinctive than for the previous AAM object types. Secondly, the objects are present in the ground-truth data at a greater range of scales as evidenced by figure 4-41 on page 126. This increase in scale variability in the ground-truth data may be of more benefit to depth based feature representations of the object parts because the ground-truth examples for this object type are especially noisy in the colour imagery.

Even though the depth based features offer improved accuracy over the original colour based features, the overall detection accuracy is still very low indicating that the features and / or the data are not sufficient to detect the objects with good reliability. As seen in figure 5-22, all of the Hough maps generated from each experimental configuration for the example ground-truth image shown in figure 5-22(a) are washed out and no strongly clustered maximum is

shown that denotes the presence of the single ground-truthed example (which is indistinct even in the colour image).

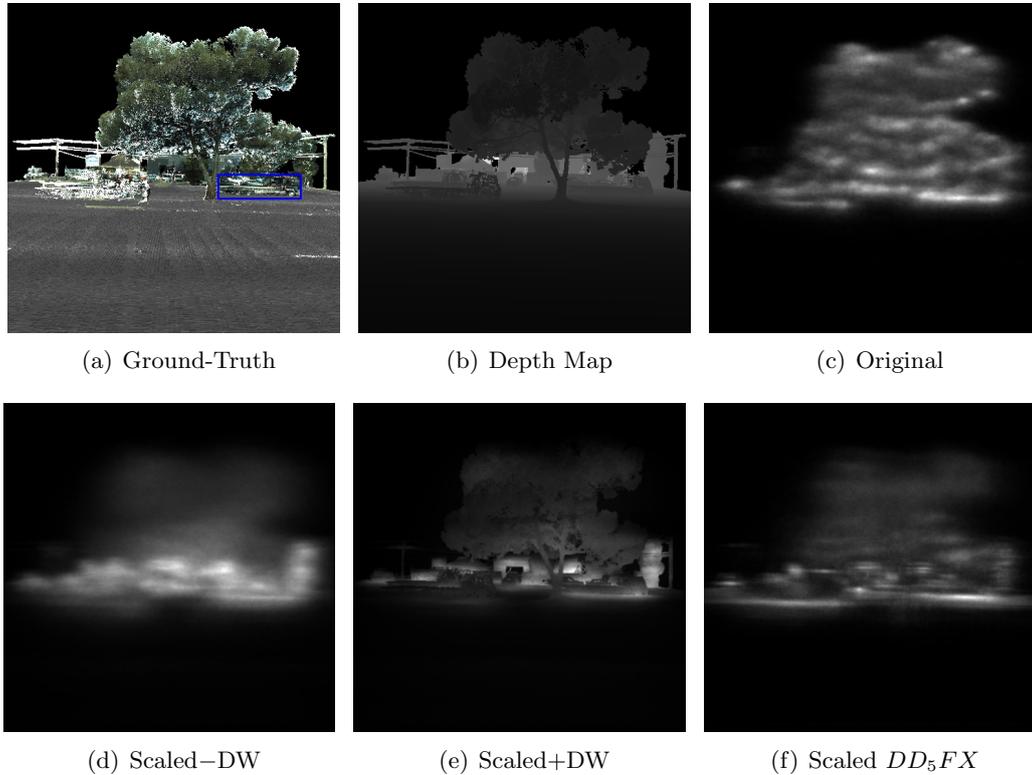


Figure 5-22: A representative query image from the AAM “Truck / Van” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.

5.4.10 AAM “Road Light”

The results for the AAM “Road Light” object class are shown in the precision-recall curves of figure 5-23. Only the depth based features are effective in improving detection accuracy (specifically precision accuracy at given recall levels) in the case of the AAM “Road Light” object class. The patch / offset scaling extension is, again, ineffective in improving object detection accuracy over the baseline results, and whether or not depth weighting is enabled has no significant bearing on accuracy. Once again, it is likely that the limited range of scales that the object type is represented at is allowing the fixed size patch descriptors used in the original CSHF configuration to encode scale dependent features about the object type causing detection accuracy to be higher than in the patch / offset scaling configurations.

For the example query image shown in figure 5-24(a), comparing the Hough vote maps gen-

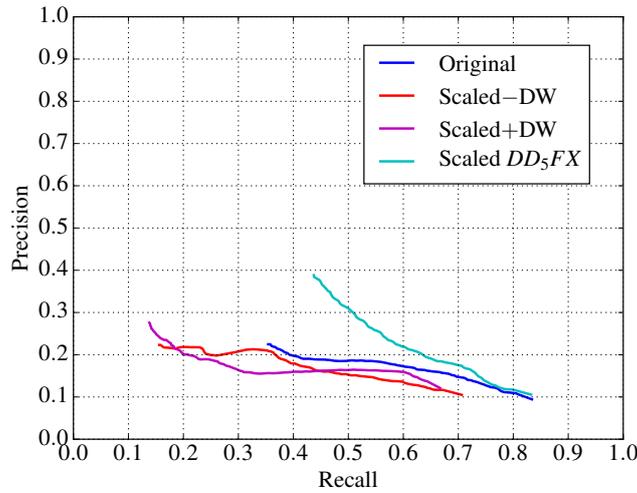


Figure 5-23: Precision versus Recall for AAM “Road Light” detection using CSHF with and without depth extensions.

erated from the original and the scaled DD_5FX experiments shown in figures 5-24(c) and 5-24(f), it is clear that the depth based features are better able to accurately locate the base of the road light. Given the very poor detection accuracy using the colour image based features, this is almost certainly due to the extremely noisy colour imagery in the AAM dataset rather than any evidence that the depth based features are intrinsically better for object detection.

5.4.11 AAM “Telegraph Pole”

Figure 5-25 shows precision-recall curves for the results from experiments carried out on the the AAM “Telegraph Pole” object type. The results for the AAM “Telegraph Pole” follow the trend for the other AAM object types tested so far. The patch / offset scaling extensions fail to improve detection accuracy. The depth-weighting extension has no practical bearing on detection accuracy even though its qualitative effects clearly differentiate the results from the two corresponding experiments (contrast figure 5-26(d) with figure 5-26(e)).

Using the example query image shown in figure 5-26(a), the improvement in precision using the depth based features is clearly demonstrated when comparing the Hough vote map generated from the baseline CSHF configuration in figure 5-26(c) with the vote map generated from the scaled DD_5FX experiment shown in figure 5-26(f). In particular, the strong false positive local maximum shown in the middle of the three maxima in figure 5-26(c), is absent in figure 5-26(f).

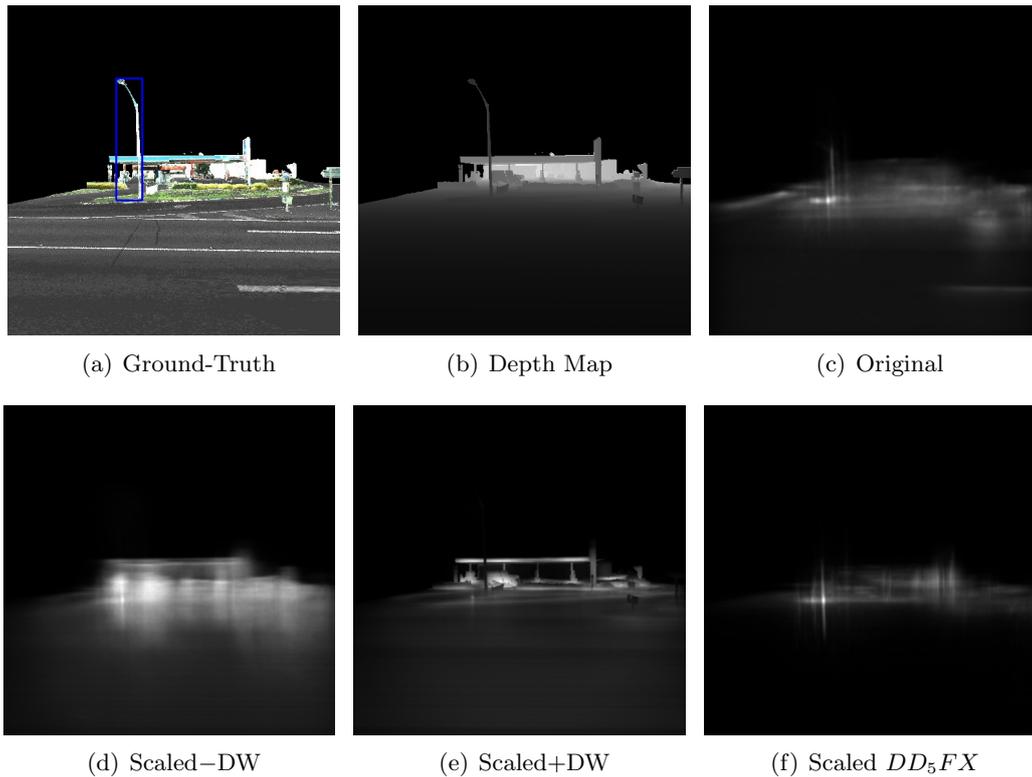


Figure 5-24: A representative query image from the AAM “Road Light” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.

5.4.12 AAM “Rectangular Road Sign”

Precision-recall curves showing the results for the experiments carried out on the AAM “Rectangular Road Sign” object type are shown in figure 5-27. listed in table 5.2 carried out on the AAM “Rectangular Road Sign” object type.

Once again, the depth extensions are ineffective in improving detection accuracy for this object type. On this occasion, the depth based features are even less effective than the colour image based features. This is likely due to the fact that examples of the object type are relatively flat in depth and have few distinguishing characteristics. In addition, examples of the object type are easily confused for other objects such as traffic lights, light poles and trees. Some examples of the generated Hough vote maps from the different experimental configurations are shown in figure 5-28.

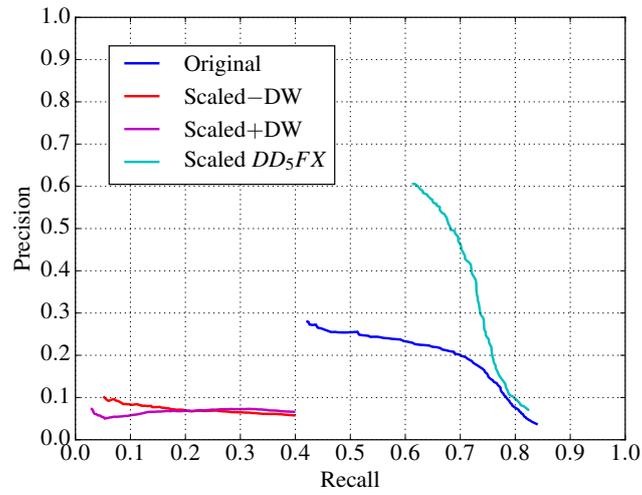


Figure 5-25: Precision versus Recall for AAM “Telegraph Pole” detection using CSHF with and without depth extensions.

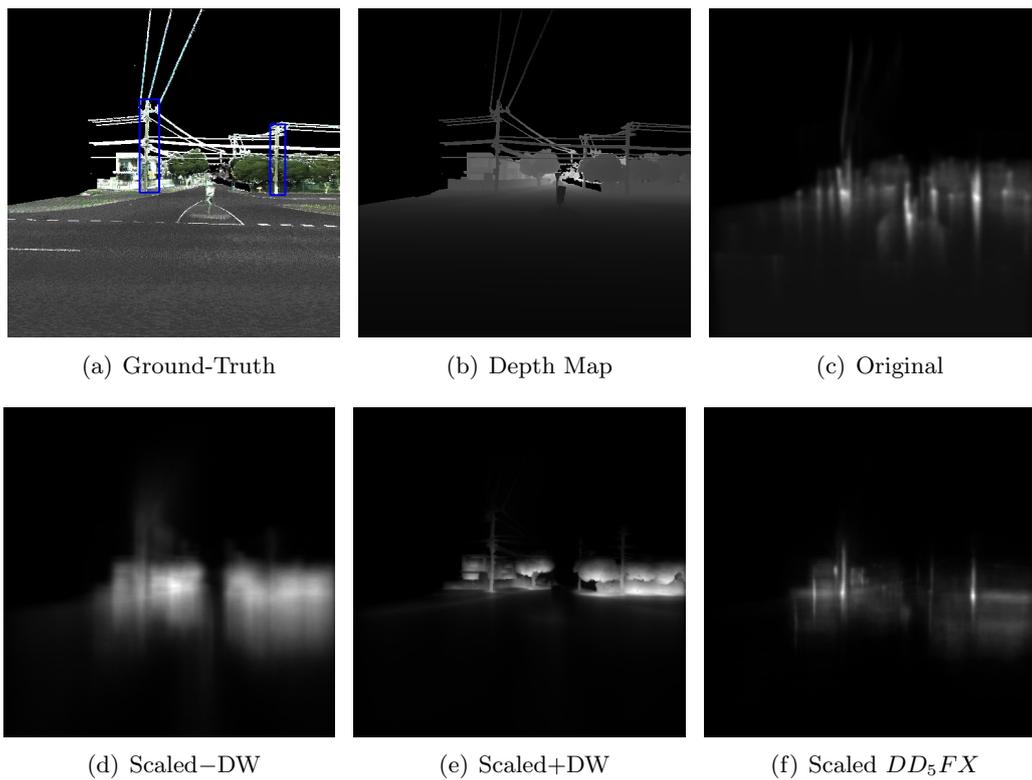


Figure 5-26: A representative query image from the AAM “Telegraph Pole” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.

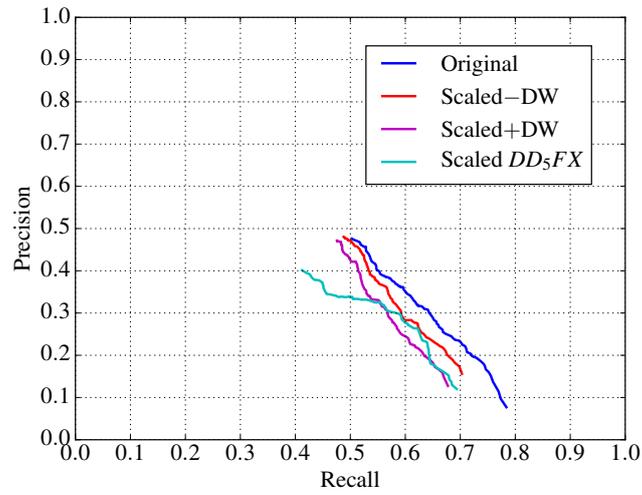


Figure 5-27: Precision versus Recall for AAM “Rectangular Road Sign” detection using CSHF with and without depth extensions.

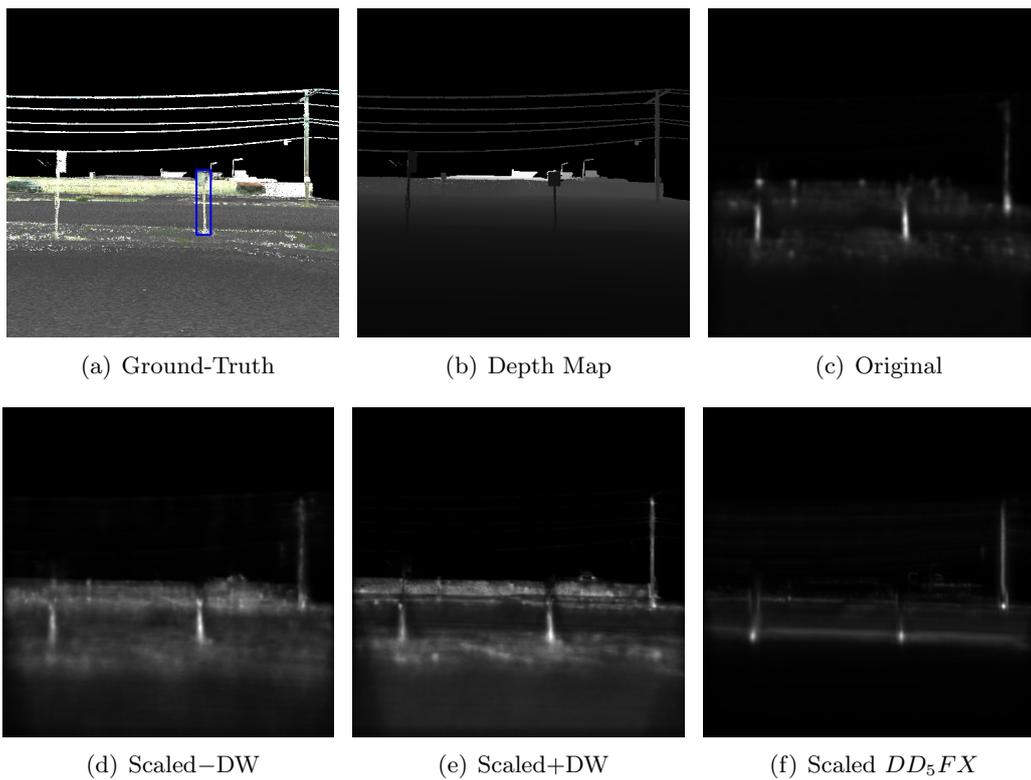


Figure 5-28: A representative query image from the AAM “Rectangular Road Sign” dataset with corresponding depth map, and generated Hough vote maps for the four different experimental configurations.

5.4.13 Overall Analysis

The results show clear differences between the detection accuracy achieved by the CSHF depth extensions using the Earthmine datasets, and the accuracy achieved using the AAM datasets. This is mainly due to the very different quality in each of these datasets. In the Earthmine data, the colour imagery is of high quality and this preferences the colour based features. In the AAM data, the depth data are of much higher relative quality and this preferences the depth based features.

In the case of the Earthmine object types, the patch / offset scaling extension largely improved detection accuracy by increasing precision at existing recall rates. That is, the extension did not help to detect objects that were not already detected by the baseline CSHF, but the extension helped to reduce the rate of false positive detections in the query images, increasing overall confidence in the existing detections.

In none of the results for the Earthmine object types did the depth weighting extension significantly improve detection precision beyond that already achieved without the use of the extension.

Earthmine’s poor quality depth data necessitated that the depth images first be preprocessed using the procedure outlined in section 5.3.2. This resulted in unavoidable corruption of any existing shape information present in depth meaning that the depth based features were not able to improve detection accuracy. For all of the Earthmine object types apart from the “Car” object type, detection accuracy was severely degraded.

Overall, the poor quality colour information in the AAM dataset meant that the colour based descriptors faired less well than the depth based descriptors. For the larger object types and the types having greater projected surface area including the “Truck / Van”, “Car”, “Telegraph Pole”, and “Road Light” types, the depth based descriptors with only ten dimensional patch descriptors outperformed the much larger colour based 8192 dimensional patch descriptors. Recall was found to be nearly as high as in the baseline configuration, but with higher precision at all confidence thresholds. It is likely that the high precision depth information increases object detection accuracy when using the depth based features. However, the reason the depth based features achieve higher levels of accuracy than the colour based features for these object types is more to do with the fact that the colour imagery is of a much lower quality in the AAM data; the colour based features are correspondingly less able to discriminate between object parts and background scene elements.

In the case of the smaller, less distinctive objects – the “Traffic Light”, “Rectangular Road Sign”, and “Triangular Road Sign” object types, the depth based descriptors are less effective. Slightly lower recall and precision levels compared to the control experiment were observed in these cases. A depth based feature is not a good candidate for encoding characteristics about objects that are essentially flat in depth (such as signs).

For the AAM object types, the patch / offset scaling extension when combined with the original colour based features of section 5.2.2 tended to give much lower detection accuracy in both recall and precision than without the patch / offset scaling extension – whether or not depth weighting was being used. This is unexpected because the higher precision depth information in the AAM datasets should mean that patches from similar relative spatial locations in an object are clustered together in the trained CSHF, resulting in improving detection accuracy. These counter intuitive results are explained when considering the much lower variation in the size of the examples in the AAM datasets, and the poor quality colour information. Any potential gains in detection accuracy that the depth extensions might offer are not realised because the baseline CSHF configuration is able to encode scale dependent features about the object types that remain accurate during validation.

5.5 Conclusion

The results on the Earthmine object types show that incorporating scale both in the determination of the patch subregions from which descriptors are extracted, and in the encoding of the object part offset vectors, is able to improve object detection accuracy using the CSHF. This improvement in accuracy is achieved without making any modifications to the underlying CSHF algorithm, and using existing colour based feature descriptors. While the results on the AAM object types do not support this conclusion, this is almost certainly due to the poor quality colour information present in this dataset. Given the higher quality colour information in the Earthmine data, as well as the fact that the patch / offset scaling extension does not require highly accurate depth information in order to be effective, it is concluded that where coregistered depth and RGB data are available, incorporating scale using the depth data in the manner described can effect large gains in object detection accuracy.

For the AAM object types, the very low dimensional depth feature descriptors described in section 5.2.2 result in detection accuracy that is at least as high as when using the original set of RGB based features. However, given the poor quality colour imagery in this dataset, the result is more reasonably explained by the inadequacy of the colour based features in being able to discriminate between parts of objects and background scene elements. This situation is

reversed in the case of the Earthmine data where the RGB based features are more effective than the depth based features due to the much higher quality colour information. Given a dataset having both accurate RGB and depth information, features generated from both of these modes may improve detection accuracy by a greater margin than by simply using features from one of these modes.

It was expected that the depth weighting extension would help to lessen the influence of false positive part detections, thereby improving precision at a given level of recall. While qualitative differences in the generated Hough vote maps were observed, this did not translate into enhanced detection precision. Overall, no significant improvement in precision was observed using this extension. The effect of depth weighting was not tested independently from the patch / offset scaling extension. However, using depth weighting without patch / offset scaling if depth information is explicitly available makes little sense given the large improvement in detection accuracy resulting from the use of patch / offset scaling.

5.5.1 Limitations

The main limitations of the experiments concerns the poor quality datasets used to evaluate the efficacy of the depth based extensions in improving object detection accuracy. Neither the Earthmine or the AAM datasets are ideally suited to this task.

Earthmine’s lower quality depth data mean that it is not possible to use the depth imagery in its native state. The preprocessing of the depth images is only sufficient to restore missing depth data under the assumption that individual parts of the same object have similar depth and that at least a single valid pixel in depth is available at the top of each object of interest. The depth based patch descriptors are unable to meaningfully encode the surface characteristics of the objects, and this is true whether or not the Earthmine depth images have been processed. This leads to the poor performance of the depth based descriptors on the Earthmine object types. The patch / offset scaling extension faired better using the processed depth images because it needs only general estimates of the depth of an object over its projected surface in order to be effective in improving detection accuracy.

Particularly poor performance was observed over most of the object types in the AAM dataset. This was due to several factors. Firstly, unlike the Earthmine data, some views in the AAM dataset are repeated several times because of the method of data collection. Panoramas were collected with fixed periodicity meaning that when the mapping vehicle was momentarily stopped (*e.g.* at traffic lights), data continued to be captured. This resulted in several

captures of the same scene and a subsequent ground-truthing of the same individual object examples – resulting in a training bias toward these examples. A more stringent ground-truthing and data cleaning process would have helped to improve the ability of the CSHF to learn more representative object types, and to improve the overall detection accuracy.

Secondly, due to the short range of the laser scanners and the larger average distance of the mapping vehicle to the objects of interest, many of the objects are not represented against visible background. Since patch descriptors are not extracted from image regions where depth information is unavailable, this results in a number of objects not having their parts used for training (or being detected).

Thirdly, the AAM data are significantly corrupted in the colour imagery resulting in degraded detection accuracy when using the colour based features. Objects closer to the image plane also have degraded depth due to the fixed sampling density of the laser scanners resulting in a “mesh-like” appearance in depth for these objects. This degrades the ability of depth based features to encode surface morphology. In general, only the objects in the AAM dataset having relatively larger exposed surface areas at larger distances from the image plane (the “Car” and “Truck / Van” object types) allow the depth based features to encode salient information about the objects.

Finally, the range of scales of the different object types in the AAM dataset is very limited. Since the depth extensions are designed to show how objects at varying scales can be detected more accurately, this lack of variability in distance within each object category makes it very difficult to produce results that meaningfully represent the potential of the CSHF’s depth based extensions. Significantly, this deficit in the AAM data adversely affects the depth extended configurations of the CSHF – especially the patch / offset scaling extended version – while acting to improve the accuracy of the original CSHF configuration since this is no longer adversely impacted by scale effects; the original CSHF models each object type at its frequently occurring scale and the encoding of features that depend on the scale of the objects is not penalised.

One possible deficiency in the patch / offset scaling approach that is unrelated to the data concerns the accuracy of voting for objects at larger scales. The placement of votes for larger objects in the query images will be less accurate because small inaccuracies are amplified at larger scales. As noted in section 5.3, these inaccuracies could be accounted for by filtering the resulting vote maps using Gaussian kernels that change in size according to distance from the camera plane. However, this would introduce significant computational overheads and given the very similar scales of the objects in the datasets (particularly in the AAM data), evaluating the possible improvement in detection accuracy this would bring should be deferred

until data are available having objects at a greater range of scales.

5.5.2 Further Research

Given the severe limitations in the datasets used to evaluate the presented depth extensions, these experiments should be repeated using datasets having both high quality depth and colour information. This may show that the CSHF depth extensions described in this chapter fair better than demonstrated in this thesis.

In the case of the experiments using the patch / offset scaling extension, the patches were constrained to be fixed in their dimensions of 0.5×0.5 metres for all of the tested object types. Future research should investigate the effect on detection accuracy of varying the dimensions of these patches (in concert with different feature types) for different types of object.

The experiments in this chapter showed how a simple depth based feature compared with the colour based features for object detection. Since the CSHF allows for combinations of different feature types (taken from different modalities), the CSHF can be trained on both types of features at once. This may result in heightened detection accuracy where depth features are more discriminative than colour based features in some cases and *vice versa*.

The depth based features presented in section 5.2.2 are parameterised on the number of surface points sampled. In the experiments, this was fixed to five, however this was based on a very limited evaluation on a few different object categories. It is likely that detection accuracy for different object types will benefit from specific parameterisations of the number of depth sampling points. The detection of different object types from datasets having more precise depth information may also benefit from a more involved sampling of each patch's depth. There is significant scope to investigate in more detail the utility of this feature extraction method.

Chapter 6

An Efficient Euclidean Distance Transform

The Euclidean Distance Transform (EDT) is an important and widely used basic image processing operation that is central to many other fundamental algorithms in computer vision, and object and pattern recognition. Tasks that require (or can benefit from) calculation of the EDT as part of their processing include watershed segmentation, morphological image filtering, object skeletonisation, determination of non-polygonal object centroids and shape measures, calculation of Voronoi diagrams (Dirichlet tessellations), Delaunay triangulation, shape matching, and shortest path computation (Rosenfeld and Pfaltz, 1968; Danielsson, 1980; Cuisenaire and Macq, 1999; Fabbri *et al.*, 2008; Bailey, 2012).

Within the field of object detection and localisation, distance transforms have been used to detect objects in query images by efficiently determining the least cost positions and deformations of the parts of deformable parts based object models (Felzenszwalb and Huttenlocher, 2005), and also for matching edge based models in query images (Borgefors, 1988; Huttenlocher *et al.*, 1993). Distance transforms have also been used in the matching of contour based features (Shotton *et al.*, 2005). Even though distance transform algorithms are not directly used as part of the object detection or classification methods presented earlier in this thesis, the algorithm is intrinsic to modern object detection methods that use parts-based models. Efficient implementations of the algorithm are therefore important to decrease the overall computational burden during object detection, freeing up computing resources for other tasks – especially where real-time object recognition is desirable.

Several algorithms that compute the exact EDT have been proposed. Many of these have super-linear or polynomial time complexity in the number of image pixels in the worst case, but express linear time complexity in the average case (Lotufo and Zampirolli, 2001; Saito

and Toriwaki, 1994; Cuisenaire and Macq, 1999). However, a large number of recently developed algorithms (particularly those based around parabola intersections and construction of Voronoi regions) give linear time worst case complexity (Breu *et al.*, 1995; Hirata, 1996; Meijster *et al.*, 2000; Maurer *et al.*, 2003; Felzenszwalb and Huttenlocher, 2004; Bailey, 2005; Krinidis, 2012; Wang and Tan, 2013). To date, there exist few studies that have empirically assessed the performance of these algorithms (Cuisenaire, 1999; Fabbri *et al.*, 2008), and the more recent algorithms have been assessed in Wang and Tan (2013). These evaluations compared the CPU run-time performance of the algorithms for a range of artificially produced binary images including random dots of varying densities and randomly located squares of different sizes, densities and angles of rotation (Fabbri *et al.*, 2008; Wang and Tan, 2013). Most of the algorithms gave good practical performance but no dynamic analysis of the computational complexity of these algorithms was conducted. In a more recent study by Wang and Tan (2013), new linear time algorithms (Felzenszwalb and Huttenlocher, 2004; Wang and Tan, 2013) were evaluated against a selection of the best performing algorithms from the previous study by Fabbri *et al.* (2008) using the CPU sequential run-time performance of each of the algorithms against the same set of artificial images as used in that study. Wang and Tan (2013) concluded that their new linear time algorithm provides faster practical run-time performance over all previous state-of-the-art algorithms tested.

In this chapter, a number of the most recent state-of-the-art EDT algorithms are evaluated against a newly presented method of computing the EDT employing several extensions that aim to significantly reduce the number of pixel iterations required to compute the EDT. The algorithms are tested against a range of different images for their dynamic computational complexity and their real-time sequential CPU performance. The new algorithm's computational complexity is linear in the worst case and is found empirically to have a smaller coefficient of computational complexity than the other state-of-the-art EDT algorithms in the average case. The new algorithm is also found empirically to be faster in real-time CPU performance than the existing state-of-the-art methods tested. As with other EDT algorithms of similar nature, the new algorithm has a constant space requirement in proportion to the square root of the input image size, it can be extended to functions of higher dimension, it can be modified to use distance metrics other than Euclidean, and it can easily be implemented to exploit concurrent processing (*e.g.* GPU) environments.

6.1 Distance Transform Metrics

A distance transform d maps a binary image $I(\mathbf{p}) \in \{\mathbf{F}, \mathbf{B}\}$ to an image where every foreground point (in \mathbf{F}) is set with the distance to its nearest background point (in \mathbf{B}):

$$d(\mathbf{p}) = \begin{cases} 0 & I(\mathbf{p}) \in \mathbf{B} \\ \arg \min_{I(\mathbf{q}) \in \mathbf{B}} \|\mathbf{q} - \mathbf{p}\| & I(\mathbf{p}) \in \mathbf{F} \end{cases} \quad (6.1)$$

where $\|\mathbf{q} - \mathbf{p}\|$ is the distance calculation between points \mathbf{q} and \mathbf{p} . The inverse of this operation is to produce the distance transform where every background point is set with the distance to its closest foreground point (*i.e.* \mathbf{F} and \mathbf{B} are interchanged in the above function to produce the inverse distance map). The normal and the inverse distance maps can be used for different purposes. As examples of differences in use, the normal distance map can be used to produce foreground object skeletonisations, or to extract shape metrics about an object of interest. The inverse distance map can be used to determine path distances to objects from arbitrary starting locations, or to assist in template matching algorithms.

Several different distance metrics can be used to compute $\|\mathbf{q} - \mathbf{p}\|$. Two of the most popular of these have been the city block (or Manhattan) distance metric (the L_1 norm), and the chessboard distance metric (the L_∞ norm). The city block distance metric assumes 4-way connectivity between pixels and distances are computed by summing over the 4-connected paths between endpoint pixels in the image:

$$\|\mathbf{q} - \mathbf{p}\|_1 = |q_x - p_x| + |q_y - p_y|, \quad (6.2)$$

The chessboard distance metric assumes 8-way pixel connectivity. In this arrangement, paths between endpoints typically give a unit cost to diagonal pixels on the path (though non-integer costs may be used).

$$\|\mathbf{q} - \mathbf{p}\|_\infty = \max(|q_x - p_x|, |q_y - p_y|). \quad (6.3)$$

For both the 4-connected and 8-connected arrangements, even though the path costs are minimised, the costs do not represent the direct (Euclidean) distance between endpoints. In the case of the city block distance metric, the path cost overestimates the straight line distance (traversing to a diagonally adjacent pixel costs 2 points). In the case of the chessboard distance metric with a unit cost set for traversing to diagonally adjacent pixels, the path cost sum underestimates the straight line distance. Examples of these two distance metrics are shown in figure 6-1 along with the straight line Euclidean distance.

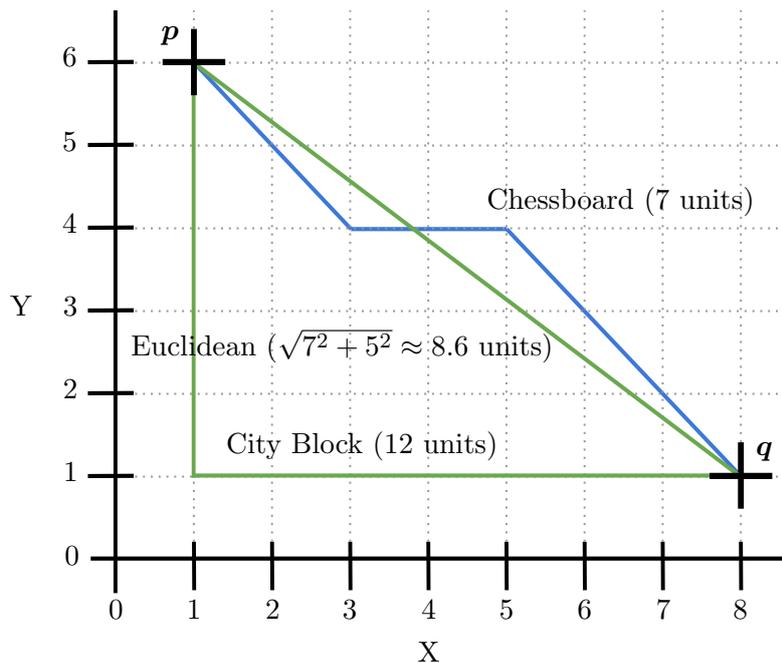


Figure 6-1: Chessboard, city block (Manhattan), and Euclidean distance metrics.

Another problem with using non-Euclidean distance metrics is the anisotropy present in the resulting distance maps; distance transforms computed using non-Euclidean distance metrics are sensitive to rotations of the input image whereas the Euclidean distance is an isotropic metric. Being rotationally symmetric, it is invariant to rotations of the input image (*i.e.* a calculated distance from point A to point B does not depend upon the angle the line segment AB makes with the base of the image). An example of the anisotropy resulting from the use of non-Euclidean distance metrics is shown in figure 6-2. The different distance transforms shown compute distances over the image to a central point. Figure 6-2(a) shows the vertical and horizontal anisotropy present when using the city block distance metric. Figure 6-2(b) shows the anisotropic effect when using the 8-way chessboard distance which has the effect of reorienting the distance error into the image's diagonals. The undesirable anisotropic effect of both these metrics can be reduced by combining them using a two pass Chamfer distance transform (Rosenfeld and Pfaltz, 1966; Borgefors, 1986). This algorithm makes two passes over the image pixels with two different 3×3 based structuring elements. At each pixel in the image, the new value is set according to the minimum (or maximum) of the values within the structuring element centred over that pixel's location. A term is added to each of the values enclosed by the structuring element, the value of the term being determined by the position in the structuring element relative to the target pixel. Typically, a constant term α is added to the values found at the adjacent vertical and horizontal positions in the structuring element, and a different constant term β is added to the values at the adjacent diagonal positions in the structuring element (typically with $\beta > \alpha$). In the case where $\alpha = \beta = 1$, the algorithm

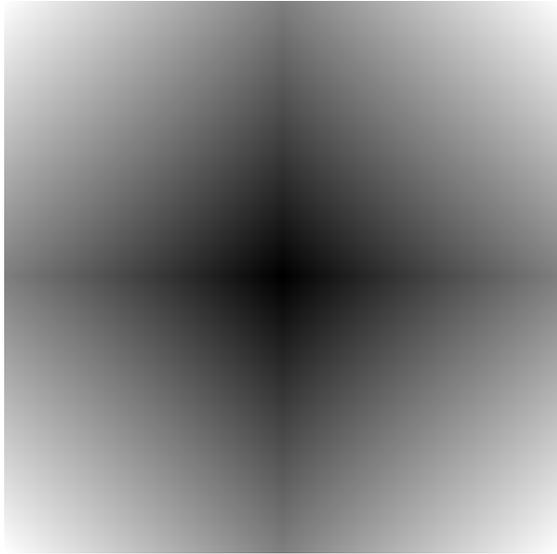
produces the same distance transform as produced when using the chessboard distance metric. In the case where $\alpha = 1$ and $\beta = \infty$, the algorithm produces the same distance transform as produced when using the city block distance metric. However, using other constant terms such as $\alpha = 3$ and $\beta = 4$ (and normalising the resulting map by $\frac{1}{4}$) allows for a distance transform to be produced that suffers with less overall anisotropy and is a closer approximation to the true Euclidean distance transform. Figure 6-2(c) shows the distance transform calculated using the Chamfer 3-4 algorithm. While larger sized structuring elements (*e.g.* 5×5 or 7×7) with different terms can produce distance transforms with progressively less anisotropy, and that are closer in value to the true Euclidean distance transform, computation times rapidly increase (Akmal Butt and Maragos, 1998). For comparison, figure 6-2(d) shows the lack of anisotropy when using the Euclidean distance metric; the distance values are the same for any point on a circle of given radius centred on the middle point in the image.

Because the Euclidean distance is invariant to image rotation, and because true straight line distance is more useful in many real world applications (such as in computing shortest paths, or for collecting shape metrics for object template matching purposes), the Euclidean distance (as given in equation 6.4) is generally preferred over other distance metrics (Bailey, 2005; Breu *et al.*, 1995; Fabbri *et al.*, 2008; Maurer *et al.*, 2003).

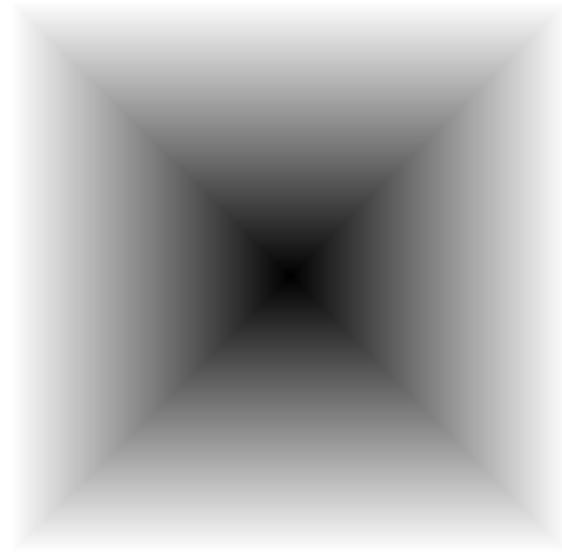
$$\|\mathbf{q} - \mathbf{p}\|_2 = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2} \quad (6.4)$$

6.1.1 Difficulties in computing the Euclidean Distance Transform

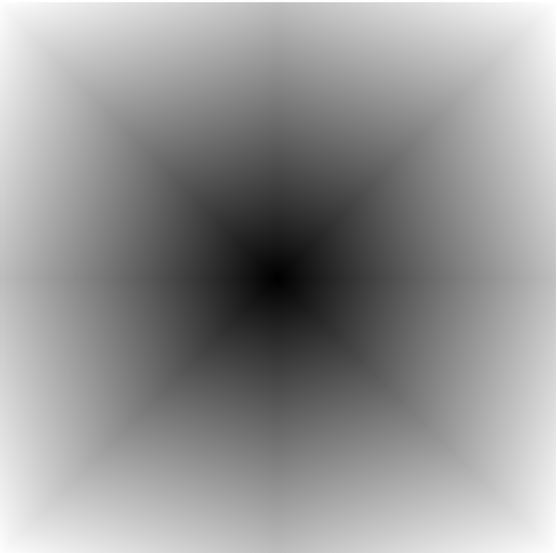
Computing Euclidean distance over discrete grids is complicated due to the necessary discretisation of continuous values. Around intersections where the distance to three or more foreground points is minimised in the continuous domain, the discrete domain does not reflect the situation of distinct, fully connected partitions being closer to their seed foreground point than any other point. This situation is otherwise known as the Voronoi diagram (or Dirichlet tessellation). In the discrete space, sometimes a single region (as given in the continuous domain) must be split into two separate regions. This problem of non-locality means that algorithms to compute Euclidean distances cannot use step counting style methods (as used for computing distance maps using the city block or chessboard distance metrics) since such methods are predicated on a discrete domain (the size of a step given by a grid square unit) that define immediately adjacent grid cells as being closer than others. Figure 6-3 describes the problem graphically. Given three grid cells ($\mathbf{g}, \mathbf{p}, \mathbf{q}$), the regions closest to them in the continuous domain are delineated by the dotted lines. However, because the discretised grid cells overlap these boundaries, selection of the closest point around the intersection of the regions (in particular, at grid cells A, B, C) is complicated. Grid cell B is disconnected from



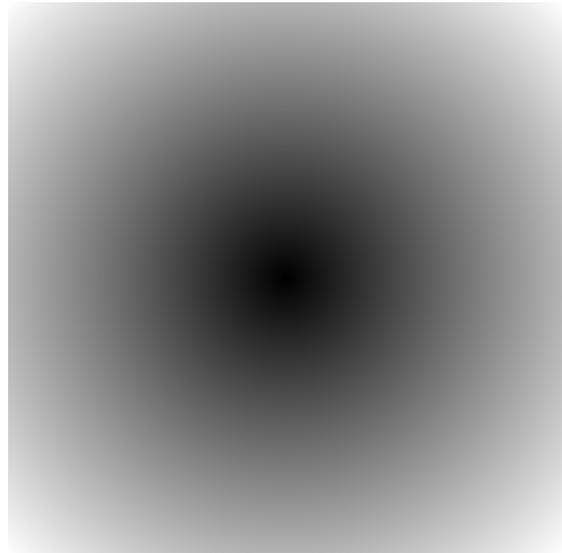
(a) Distance transform using the city block metric.



(b) Distance transform using the chessboard metric.



(c) Distance transform using Chamfer 3-4.



(d) Distance transform using the Euclidean metric.

Figure 6-2: Applying distance transforms using different metrics to an image with a single, centred feature pixel. The city block (a), chessboard (b) and Chamfer 3-4 (c) metrics produce results suffering from anisotropic effects. Only the Euclidean distance metric (d) results in an isotropic (rotationally invariant) distance map. (Images are contrast stretched for clarity.)

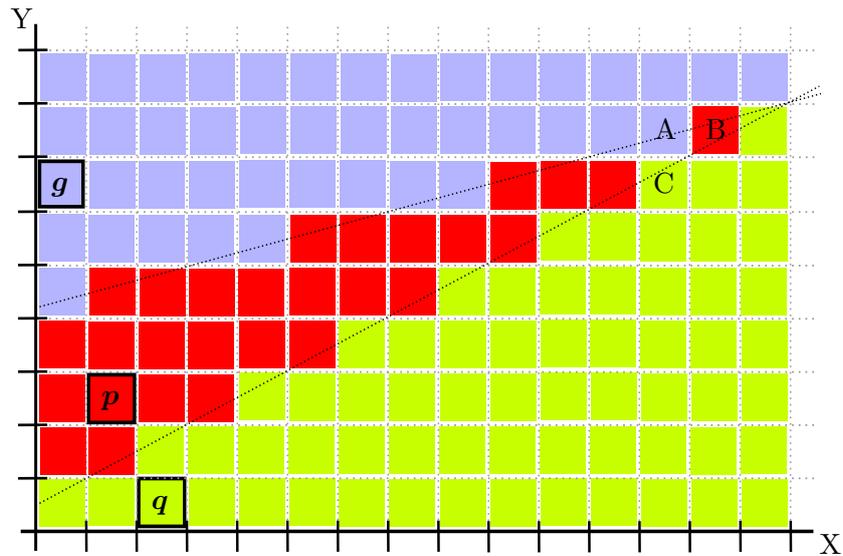


Figure 6-3: Discretisation of Euclidean distances leads to problems of non-locality.

the rest of the region that is also closer to grid cell p than to either g or q . At both A and C , the Euclidean distance to p is larger than for g and q respectively. But unlike the case for discrete distance metrics such as the city block and chessboard metrics, this does *not* imply that cell B (which is even further away in the continuous domain from g, p and q) can only be closest to either g or q . This inherent difficulty in calculating the exact Euclidean distance transform, and the practicable suitability of distance transforms produced using simpler step aggregation type algorithms and distance metrics specifically designed to be used over discrete grids has meant that fast and exact methods of computing the distance transform using the Euclidean distance metric have only recently (since the 1990s) been developed (Fabbri *et al.*, 2008).

6.1.2 Euclidean Distance Transform Algorithms

Given an image with N pixels having k background pixels (and $N - k$ foreground pixels), the number of brute force operations to compute the distance map is $k(N - k)$ (each background pixel must be tested against every foreground pixel). This expression is maximal when $k = \frac{N}{2}$. By simple substitution, the asymptotic maximum for the complexity of the brute force algorithm is therefore $O(N^2)$. The asymptotic lower bound is given when $k = 1$ (the distance transform is undefined when there are no background pixels), giving a minimum algorithmic complexity of $\Omega(N)$.

For large N the brute force $O(N^2)$ algorithm is generally too inefficient and so algorithms

that exploit properties of the distance calculation within local regions must be exploited. These local results are then propagated to other areas of the distance map to produce the final transform. The various methods by which the distance transform is calculated therefore differ in how results about local distances are calculated and propagated.

All state-of-the-art EDT algorithms actually calculate the squared EDT (the L_2^2 norm). This is because the squared Euclidean distance is always a whole integer over discrete grids of pixels and real valued representation issues (which are possible sources of error) can be avoided. Moreover, distance calculations can be sped up by using integer division and avoiding the requirement to calculate square roots. For practical use in optimisation style problems, the L_2^2 norm is usually suitable (or even preferred) in place of the L_2 norm. For applications requiring the actual Euclidean distance, the square root of the final distance map can be taken.

Fabbri *et al.* (2008) give three different categories of EDT algorithm. In the first category, *ordered propagation* algorithms (otherwise known as *grass-fire* algorithms) proceed by propagating local distance values from seed foreground/background points. While the city block and chessboard metrics are well suited to this type of algorithm (by way of counting steps), Euclidean distances can be calculated by incorporating a “bucket” list for the pixels on the propagation front where each bucket holds references to pixels having equal distance value. Although ordered propagation algorithms can be designed to run very efficiently, they are not easily extensible to functions or images having more than two dimensions, and are not straight-forwardly modified to utilise independent processing architectures. Cuisenaire (1999) gives several different algorithms to compute the EDT by propagation. One of these methods (propagation by multiple neighbourhoods – PMN) works by aggregating approximately calculated distance transform maps constructed over coarse pixel neighbourhoods. However, the method cannot (without adjustment) accommodate images of arbitrarily large dimensions and still guarantee exact EDT calculation. The algorithm has been empirically evaluated as having linear worst case complexity (Fabbri *et al.*, 2008).

Raster scanning algorithms are the second category of EDT algorithms and these operate by scanning the image in some predefined order (*e.g.* left-to-right, top-to-bottom in one pass, then right-to-left, bottom-to-top in a second pass) and typically use 2-D local region structuring elements centred on the current pixel of interest. The Chamfer 3-4 algorithm is an example of this method (though it does not compute the exact distance transform). Raster scanning algorithms that produce the exact Euclidean distance map can be fast when scanning the image, but because of the need to compute the exact distance, several different structuring elements must be used and parsed at each position in the image. This means that the same pixels must be interrogated several times over and this increases the amount of time

spent in each pixel iteration – slowing the algorithm overall. There have been no comparative evaluations of raster scanning algorithms in the recent surveys of Fabbri *et al.* (2008) and Wang and Tan (2013). Algorithms of this type include those proposed by Danielsson (1980), Shih and Wu (2004), and Krinidis (2012) who present an $O(MN)$ 2-D image raster scanning algorithm that extends the work of Danielsson (1980) to use four different masks of reduced size (where M is the combined size of the structuring elements).

Independent scanning (or *dimensionality reduction*) algorithms (Rosenfeld and Pfaltz, 1968) represent the third (and most common) category of EDT algorithms. These algorithms work over 1-D arrays of image rows/columns independently of other rows/columns in the image. As such, the locality of pixels being exploited is constrained to single dimension pixel windows centred on the current point of interest. These algorithms proceed in two phases by first parsing an image’s columns/rows to calculate the squared Euclidean distances from foreground points along one dimension, before using these results in the second phase to calculate the orthogonal row-wise/column-wise squared Euclidean distances along the second dimension. This type of algorithm has the advantage of being extensible to data of more than two dimensions because the results from previous dimension parses are used as input to the next (higher) dimension. In addition, independent scanning algorithms can be easily written to take advantage of concurrent processing environments because only individual rows/columns are written to at any one time, and the original binary input image can be left untouched in memory. Thirdly, because the algorithm is decomposed into functions on single dimension arrays, this category of EDT algorithms is conceptually simpler to understand and develop.

Independent scanning algorithms have developed along two major lines with a third, less popular approach based on the mathematical morphology of an image (erosions and dilations) (Fabbri *et al.*, 2008). The third approach, originally developed by Shih and Mitchell (1992) has been extended to enable a reduction in size of the 3×3 structuring element to a single dimension 1×3 array allowing the algorithm to work over rows/columns independently of each other (Lotufo and Zampiroli, 2001). The two more popular approaches are based around the determination of the intersections of the quadratic distance function parabolas (Saito and Toriwaki, 1994; Hirata, 1996; Meijster *et al.*, 2000; Felzenszwalb and Huttenlocher, 2004), and the calculation of Voronoi region intersections (Breu *et al.*, 1995; Ogniewicz and Kübler, 1995; Guan and Ma, 1998; Maurer *et al.*, 2003; Bailey, 2005; Wang and Tan, 2013). While Felzenszwalb and Huttenlocher (2004) cites Breu *et al.* (1995) and Maurer *et al.* (2003), their work constructs Euclidean distances by way of calculating parabola intersections. However, it can be shown that both approaches are mathematically equivalent (see section 6.2.2) and that all of these algorithms are very similar in their second stages with differences generally given in the choice of which calculations to optimise. Felzenszwalb and Huttenlocher (2004) give an $O(N)$ independent scanning algorithm that proceeds by constructing the lower bound of

the squared Euclidean distance parabolas in the input array. The lower bound is stored in a list which is finally parsed to calculate the minimum distances at each index. Wang and Tan (2013) developed a very similar algorithm to construct the lower bound of the squared Euclidean distance parabolas but derived their algorithm from the calculation of intersecting Voronoi regions. The main improvement over the work of Felzenszwalb and Huttenlocher (2004) is to optimise the calculation of intersections resulting in a faster overall run-time even though it typically requires at least as many pixel iterations and updates to the final distance map. The implementation is also made more efficient by updating the final distance map in-place requiring only minimal extra memory (proportional to the size of an image row). Wang and Tan (2013) also provide an empirical evaluation of their method compared with the method of Felzenszwalb and Huttenlocher (2004) and a selection of the best performing algorithms as evaluated in the survey of Fabbri *et al.* (2008).

This chapter presents a new algorithm of this class, also based on the method of calculation of quadratic parabola or Voronoi region intersections and derived from the earlier body of work most recently exemplified in the algorithms of Felzenszwalb and Huttenlocher (2004) and Wang and Tan (2013). In this new method, a set of algorithmic optimisations are introduced to allow certain calculations and updates to the distance map to be avoided entirely while the computational optimisations given previously by Wang and Tan (2013) are retained. These new extensions result in an algorithm that, in many cases, computes the EDT more than twice as quickly as previous methods and is generally faster and more efficient (in computational complexity) than the previous state-of-the-art EDT algorithms. Though not as stable, in some cases, the new algorithm even compares in speed to the Chamfer 3-4 algorithm which has the potential to increase accuracy in applications that might benefit from exact distance calculations but are currently too resource constrained to practically implement existing methods.

6.2 The Extended EDT Algorithm

The new extensions are explained in the context of a generalised independent scanning algorithm of the kind implemented by Felzenszwalb and Huttenlocher (2004) and later by Wang and Tan (2013) with specific references made to those algorithms where relevant. The main advantages proffered by the new extensions are to avoid making unnecessary updates to the input/output distance map, and at each point in the image, to reduce the number of comparisons with members of a list of potentially nearer points.

As described here, the algorithm actually calculates the inverse Euclidean distance transform (*i.e.* each background point is set with the distance to its nearest foreground point). Like all independent scanning algorithms, the algorithm proceeds in two stages. In stage one, pixels are scanned in columns (or rows) to calculate the minimum one-dimensional vertical (or horizontal) distance from each foreground pixel in the column/row to every background pixel in the column/row. The actual distance metric calculated depends on the input required to the second stage of the algorithm, however in the implementation by Felzenszwalb and Huttenlocher (2004), the distance metric is the squared Euclidean distance. In the method of Wang and Tan (2013), this stage calculates the squared Euclidean distance of each nearest foreground point in the column/row to the image origin (*i.e.* the top left corner), rather than to the background point itself. This is to facilitate the efficient computation of the parabola or Voronoi region intersections (see section 6.2.2). In the algorithm of Felzenszwalb and Huttenlocher (2004), no separate stage one algorithm is given because the stage two algorithm actually subsumes the logic of stage one (and the method is presented in the context of performing distance transforms on arbitrary 1-D arrays). The stage one algorithm as used by Wang and Tan (2013) for the case of a 2-D (or higher-D) image is presented here. This algorithm is more efficient (requiring $\Omega(n)$ array accesses/updates in the best case) than the $\Omega(2n)$ algorithm used by Lotufo and Zampirolli (2001); Maurer *et al.* (2003) and others (Fabbri *et al.*, 2008) (where n is the length of a row/column image array).

6.2.1 First Stage Calculation

The pixels in each column are independently scanned top to bottom with squared Euclidean distances to the last parsed foreground pixel updated until a new foreground pixel is seen, whereupon the point of equidistance t with the previously seen foreground point is calculated, and updates proceed back in the opposite direction. Values in the array up to this point t are overwritten with the squared Euclidean distance to the new foreground pixel and when point t is reached, iteration picks up again where it left off, proceeding once more in the

original direction, updating values to the right of the most recently seen foreground pixel. Pseudo-code for this stage one column-wise (row-wise) scan is given in algorithm 2. For a column/row of length n , the complexity of this algorithm is $O(n)$. Specifically, in the worst case, the maximum number of value revisions (updates in the opposite direction governed by the inner while loop) is $n - 1$ and this happens when the input array has only a single foreground point located at the end of the array. In this case, the algorithm is no faster than the simple $\Omega(2n)$ method given by Lotufo and Zampiroli (2001).

Algorithm 2 EDT Stage 1: Fast in-place calculation of 1-D squared Euclidean distances

```

1: procedure CALC1DSQUARED EUCLIDEAN DISTANCE(array)
2:    $dsum \leftarrow \infty$ 
3:    $t \leftarrow -1$ 
4:   for  $i \leftarrow 0, \text{LENGTH}(\textit{array})$  do
5:     if  $\textit{array}[i] = 0$  then
6:       if  $t > -1$  then ▷ min index to iterate back to when updating distances
7:          $t \leftarrow (t + i)/2$ 
8:       end if
9:        $d \leftarrow 1$ 
10:       $dsum \leftarrow 1$ 
11:       $k \leftarrow i - 1$ 
12:      while  $k > t$  do ▷ Update previously set larger distances
13:         $\textit{array}[k] \leftarrow dsum$ 
14:         $k \leftarrow k - 1$ 
15:         $d \leftarrow d + 2$ 
16:         $dsum \leftarrow dsum + d$ 
17:      end while
18:       $t \leftarrow i$  ▷ Update last seen index of foreground pixel
19:       $d \leftarrow 1$ 
20:       $dsum \leftarrow 1$ 
21:    else
22:       $\textit{array}[i] \leftarrow dsum$  ▷ Set distance to current closest foreground point
23:       $d \leftarrow d + 2$ 
24:       $dsum \leftarrow dsum + d$ 
25:    end if
26:  end for
27: end procedure

```

6.2.2 Second Stage Calculation

Assuming a column-wise scan of the image in stage one, in stage two, the image's rows are independently parsed and each array location in turn is compared against a list of previously parsed row locations stored as a stack. The value of each array element is the squared

Euclidean distance to its nearest orthogonally located foreground point (either above or below the row). As iteration proceeds from left to right along the row, the row intersection of each new point with the previously seen point is calculated. The orthogonal distance to each foreground point is the squared distance for the ordinate of the foreground points, hence this calculation is equivalent to equating the two parabolas representing the squared Euclidean distance functions rooted at the (abscissae, $value^2$) of the corresponding elements in the row array. This derivation of the intersection is used by Felzenszwalb and Huttenlocher (2004) and is depicted in figure 6-4. The figure shows the pixel rows and columns of an image with two foreground points \mathbf{p} and \mathbf{q} . Their respective vertical distances from the row currently being parsed (row 0) are 1 and 2. The point along the current row t that is equidistant from these points is given by equation 6.5 (also shown on the diagram) as the intersection of the corresponding blue parabolas.

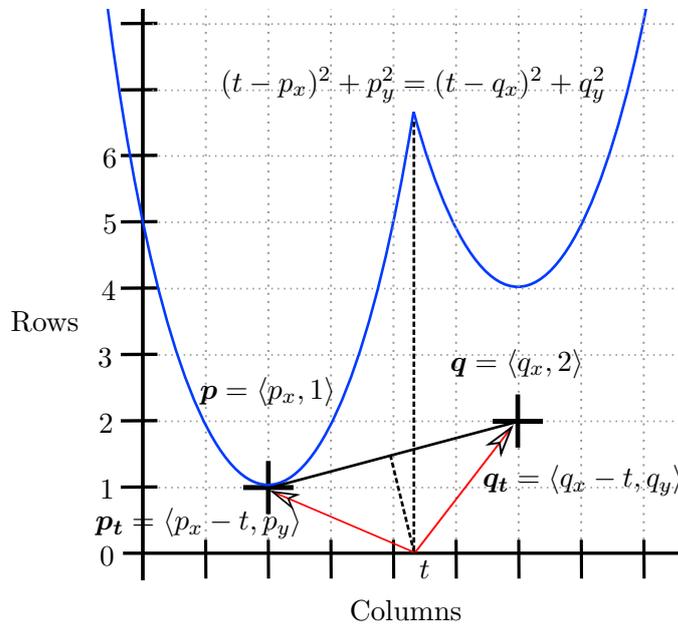


Figure 6-4: Calculation of the location of intersection along the row being equidistant from foreground points \mathbf{p} and \mathbf{q} by parabola intersection (Felzenszwalb and Huttenlocher, 2004)

The derivation of this intersection location t is then given by:

$$(t - p_x)^2 + p_y^2 = (t - q_x)^2 + q_y^2 \quad (6.5)$$

$$t^2 - 2tp_x + p_x^2 + p_y^2 = t^2 - 2tq_x + q_x^2 + q_y^2 \quad (6.6)$$

$$2t(q_x - p_x) = (q_x^2 + q_y^2) - (p_x^2 + p_y^2) \quad (6.7)$$

$$t = \frac{(q_x^2 + q_y^2) - (p_x^2 + p_y^2)}{2(q_x - p_x)} \quad (6.8)$$

$$p_x^2 - 2tp_x + t^2 + p_y^2 = q_x^2 - 2tq_x + t^2 + q_y^2 \quad (6.12)$$

$$2t(q_x - p_x) = (q_x^2 + q_y^2) - (p_x^2 + p_y^2) \quad (6.13)$$

Equations 6.7 and 6.13 are the same, so these methods are clearly equivalent. Letting p_h be the squared Euclidean distance of \mathbf{p} from the origin (*i.e.* $p_h = p_x^2 + p_y^2$), then

$$2t(q_x - p_x) = q_h - p_h \quad (6.14)$$

$$t = \frac{q_h - p_h}{2(q_x - p_x)} \quad (6.15)$$

By pre-calculating p_h during stage one, Wang and Tan (2013) are able to optimise the calculation of t (which is called very frequently) leading to a much faster algorithm for the same number of iterations. q_x is always strictly greater than p_x , hence the sign of t is determined by the numerator of equation 6.15. This is particularly important to note for implementation in C/C++ where division of negative values is *truncated towards zero* which can lead to incorrect calculation of t in rare cases. This possible source of error can be avoided because it is only necessary to know that a negative value of t will occur (entailing that the previously stored point be popped from the candidate nearest points stack). This test also allows for the use of integer division in equation 6.15 leading to a further speed up. All of these optimisations are implemented by Wang and Tan (2013).

A more intuitive derivation of t can be seen in figure 6-5 when one considers that the right triangle constructed from vector vertices $\mathbf{A} = \langle \frac{q_x + p_x}{2}, \frac{q_y + p_y}{2} \rangle$, $\langle t_{\mathbf{qp}}, 0 \rangle$, and $\langle A_x, 0 \rangle$ is similar to the right triangle constructed from vector vertices \mathbf{p} , \mathbf{q} , and $\langle p_x, q_y \rangle$ (the two blue triangles in the figure) but rotated a quarter circle and scaled. The intersection t can then be derived simply by undoing the scale and rotation: setting the ratios of the sides of these triangles inversely equal to one another and multiplying through.

$$-\frac{q_y - p_y}{q_x - p_x} = \frac{2}{q_y + p_y} \left(\frac{q_x + p_x}{2} - t \right) \quad (6.16)$$

$$p_y^2 - q_y^2 = q_x^2 - p_x^2 - 2t(q_x - p_x) \quad (6.17)$$

$$t = \frac{q_h - p_h}{2(q_x - p_x)} \quad (6.18)$$

Once calculated, while $t_{\mathbf{qp}}$ is at least as small as the previously calculated and stored $t_{\mathbf{pg}}$ (*i.e.* $t_{\mathbf{qp}} \leq t_{\mathbf{pg}}$), \mathbf{p} is popped from the stack and new intersection location $t_{\mathbf{qg}}$ is calculated and set as the left bound of \mathbf{q} with \mathbf{g} . This is the calculation of the location of intersection for the Voronoi regions along row R having \mathbf{q} and \mathbf{g} as their closest points. Point \mathbf{p} can be discarded because its distance to any location in row R is never less than the minimum distance at any location in the row to either the point to its left (\mathbf{g}) or the newly parsed point to its

right (q). This evaluation of previously stored points on the stack against new point q may take several iterations and it continues until the calculated intersection between q and an old point on the stack is found to be at least as large as the stored intersection for that point and the immediately nearest point to its left (next on the stack). When t_{qp} is found to be strictly greater than the intersection of p and the previous nearest point on the stack g (i.e. $t_{qp} > t_{pg}$), q (and its left bound intersection with p) is pushed onto the stack. This process is depicted in figure 6-6 in the case of a single iteration (comparison of new point q against previously stored candidate nearest points g , and p immediately to its left).

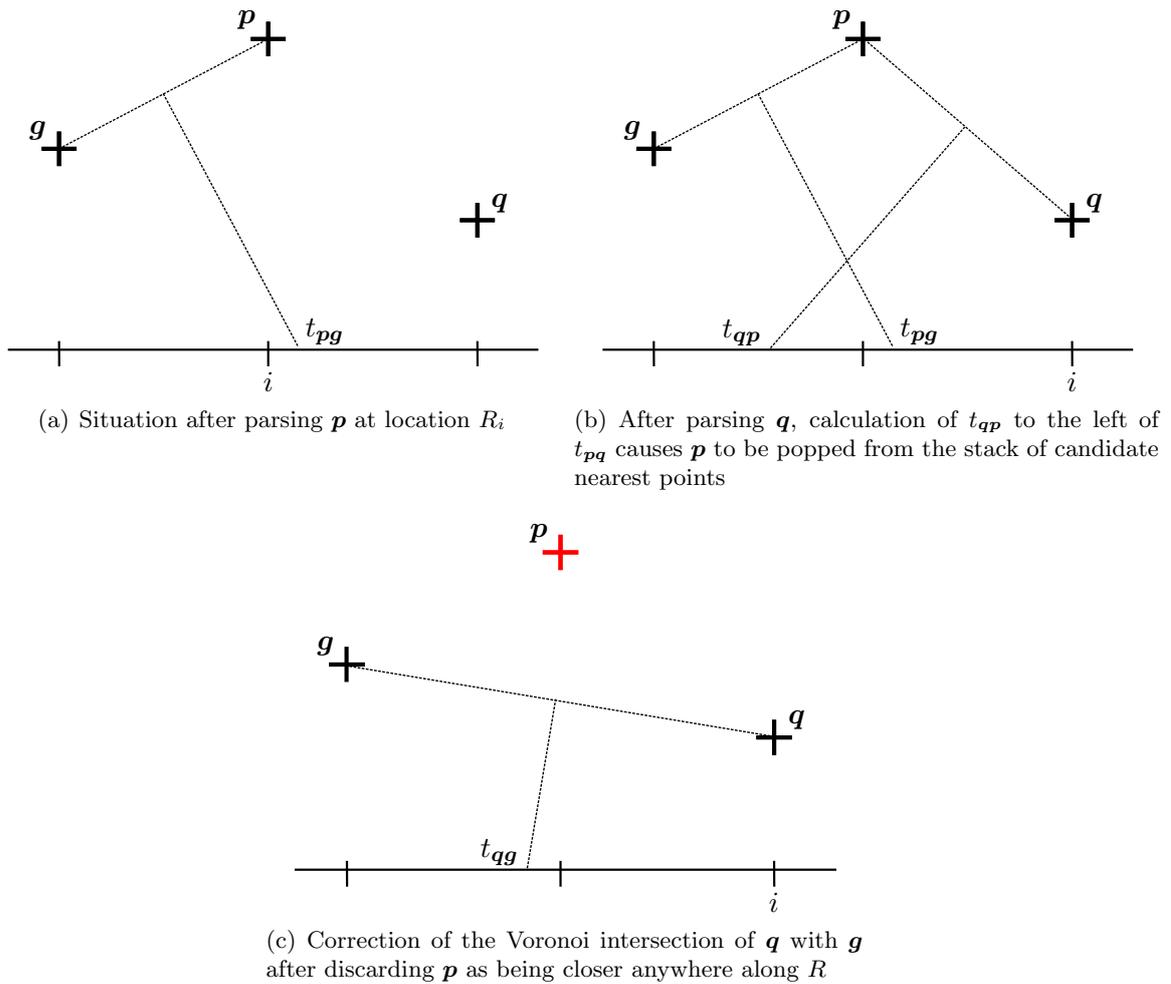


Figure 6-6: Process of comparing the intersection of the Voronoi region at row R for new point q against previously stored points g and p (iteration along R at i in each diagram moving left to right)

In stage three of the algorithm, the completed list of nearest points is parsed and the respective row array values updated to the squared Euclidean distance to the corresponding points.

All points in the row must be parsed at least once, hence the run-time performance of this algorithm depends on the number of times candidate nearest foreground points are evaluated in the stack, and the number of updates needed to transform the original input row into the squared Euclidean distances to the final list of nearest points.

6.2.3 Algorithm Extensions

In the method described above, there are two opportunities during stage two to reduce the number of computations. Firstly, the number of comparisons to candidate nearest points on the stack can be reduced. Secondly, the number of final updates required to produce the output distance map can be reduced. The extensions described below lead to efficiency gains in both of these areas.

First Extension

The first extension stems from the fact that at commencement of the second stage, array locations that are no more than one unit distant from their orthogonally located foreground points do not need their distance values updating in the final stage three of the algorithm (*i.e.* locations which either are themselves a foreground point, or locations that are immediately adjacent to a foreground point either above or below them). For these row locations, it is impossible for any other foreground point referenced from any other location in the array to be closer. This implies that only the endpoints of contiguous regions of this nature (subsequently termed “foreground” regions) have any influence in the calculation of distances at any other locations in the array and so only these endpoints need to be added to the candidate nearest points stack.

Given a mechanism to identify foreground region endpoints during the final stage three update, points within the foreground regions (including the endpoints themselves) can be ignored. This allows the number of comparisons to entries in the candidate nearest points stack to be reduced in two ways. Firstly, the overall number of candidate points added to the stack is greatly reduced because points between the foreground endpoints are ignored. Secondly, upon identification of a foreground region point (whether it is a single point or contiguous with others to its right), the whole stack of previously stored candidate nearest points can be discarded. This is because no previously stored point can be closer to any point to the right of the newly parsed point than the newly parsed point itself.

This situation is shown in figure 6-7 where the dotted lines slanting down and to the right denote the four left bound intersection locations with the current row of the Voronoi regions closest to the four points shown in red in columns h through k . All of these intersections are beyond index k . As iteration proceeds from left to right from column h to column k and each new foreground point is encountered, each is in turn pushed onto the candidate nearest points stack. When the foreground point in column l is seen (shown in blue), the candidate points stack is reset because none of the four previously parsed points having left bound intersection locations $\geq l$ can be closer than the new point at *any* location in the row.

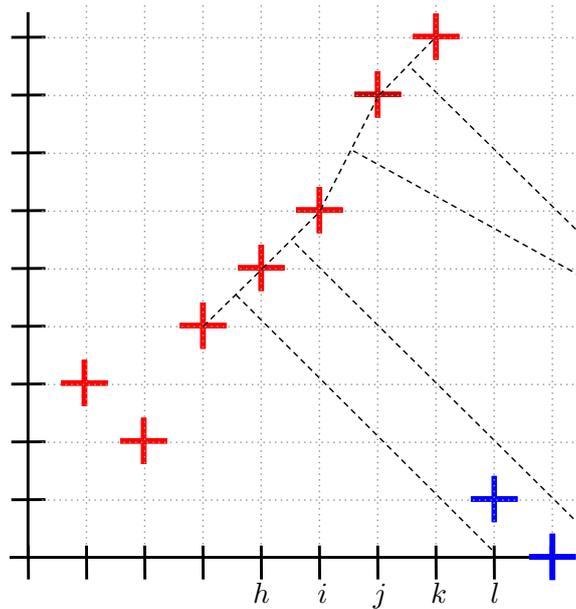


Figure 6-7: When the first foreground point at l is parsed, previously stored candidate points at locations h , i , j and k are discarded.

Further, because foreground regions can be present at either end of the row, these points can be excluded before the main loop of stage two is entered meaning that the effective length of the row is reduced.

Second Extension

The second optimisation incorporated into the algorithm immediately discards newly parsed point q after calculation of its Voronoi region intersection with the most recently stored previous point p if either:

1. The calculated intersection t_{qp} is to the left of the start or to the right of the effective end of the row, or

2. In the current row y there already exists at location $\langle t_{qp}, y \rangle$ a foreground point $\mathbf{g} = \langle t_{qp}, g_y \rangle$ such that $|g_y| \leq |\mathbf{q} - \langle t_{qp}, y \rangle|$.

This early discard of points has the effect of further reducing the overall size of the candidate nearest points stack, and thus the upper bound on the number of comparisons when parsing points further along in the row.

Figure 6-8 describes this situation graphically. The row intersection locations of red points \mathbf{p} and \mathbf{q} with previously stored closest point \mathbf{g} (t_{pg} and t_{qg}) are both within the array bounds (the right edge of this figure) but the orthogonal distances from row y at these locations to blue points \mathbf{u} and \mathbf{v} are at least as short. Therefore, points \mathbf{p} and \mathbf{q} can be immediately discarded instead of being pushed onto the candidate nearest points stack.

In the case of points \mathbf{r} and \mathbf{s} , their calculated row intersections are beyond the effective end of the array, and so they will never be closer at any point along the row than point \mathbf{g} ; these points too can be discarded immediately upon parsing.

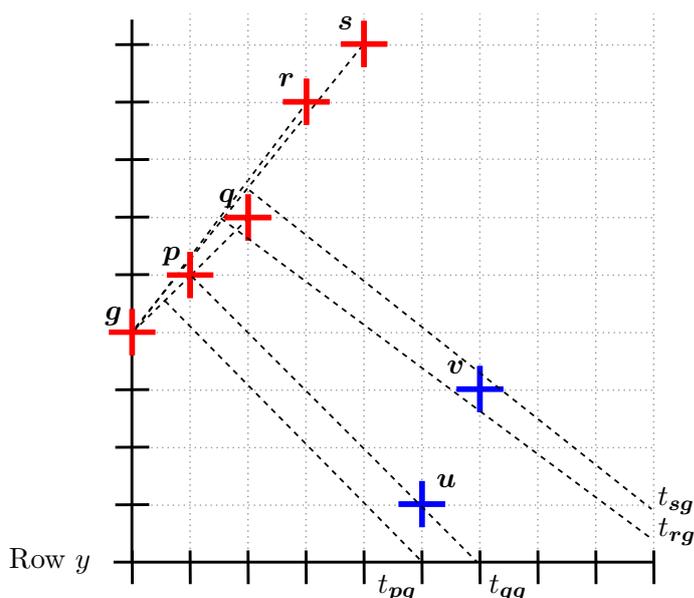


Figure 6-8: A newly parsed point \mathbf{p} is discarded early if its location of intersection t along row y is either beyond the effective end of the array, or the orthogonal distance from row y to a foreground point already in column t is $\leq |\mathbf{p} - \langle t, y \rangle|$.

All of the previous optimisations of Wang and Tan (2013) are retained. In particular, instead of calculating p_h in-place during stage one of the algorithm (requiring that all array values be updated in stage three), p_h is calculated during the stage two identification of nearest points and stored as part of the candidate points list. For these purposes a small struct (a

dedicated collection of ordered variables in memory) is used to collate candidate point values. Letting $\mathbf{p} = \{x, d, b, h\}$, p_x is the ordinate of \mathbf{p} , p_d is the value at $R[p_x]$ (the orthogonally row offset squared Euclidean distance of the point as calculated during stage one), p_b is the left exclusive array bound of the region of the row closest to \mathbf{p} (given by the integer division calculation of t), and $p_h = p_x^2 + p_d$ (calculated once and stored for reuse). Calculating p_h during stage two of the algorithm is no more costly than computing it during stage one, and this allows updates to large sections (all of the foreground regions) of the distance map to be avoided entirely because the original input array is left untouched. In the best case where foreground regions occupy all or the majority of the image, the required total number of pixel iterations tends toward $\Omega(2N)$. The previous minimum number of pixel iterations required (as in Felzenszwalb and Huttenlocher (2004) and Wang and Tan (2013)) was $\Omega(3N)$ because of the need to calculate squared Euclidean distance values for every entry in the input array.

For implementation convenience, two macros are defined. Let $\text{calct}(q_x, q_h, p_x, p_h)$ be defined according to equation 6.15 using integer division as algorithm 3. For readability, let

Algorithm 3 Calculation of the location of row intersection using integer division

Ensure: $q_x > p_x$
function CALCT(q_x, q_h, p_x, p_h)
 if $q_h < p_h$ **then**
 return -1
 else
 return $\frac{q_h - p_h}{2 * (q_x - p_x)}$ ▷ Never negative (see equation 6.15)
 end if
end function

$\text{SQDIST}(\mathbf{q}, i) = (q_x - i)^2 + q_d$, *i.e.* the squared Euclidean distance of foreground point \mathbf{q} from row array location i . Pseudo-code for stage two is given in algorithm 4.

The function described by listing 5 is called prior to entry of the main loop in algorithm 4. This function skips over the contiguous portion of points in the row (starting from the left) that are too far away to be considered in the distance transform. If no such values are found, foreground points are skipped over reducing the effective length of the row from the left. The function returns the starting index i for iteration in the main loop of algorithm 4 together with the minimum effective index of the row.

Procedure 6 is called from the main loop of algorithm 4 to compare a newly parsed point \mathbf{q} against the existing stack of candidate nearest points $npList$. The procedure updates the reference values $npList, pid_x, plast$ so that on return, \mathbf{q} has either been discarded according to the second optimisation (see line 13), has replaced the stack entirely (see line 9), or has been

pushed onto the stack at a location according to correct calculation of the Voronoi intersection of q with a previously stored point (entailing that zero or more previously stored points are popped from the stack) (see line 15). On return, $pidx$ and $plast$ are updated to point to the currently closest point and the last point in $npList$ respectively.

Finally, the previously trivial third stage update of the row array to calculate the squared Euclidean distance values is modified to recognise the endpoint pairs of foreground regions so as to ignore the array values that do not need updating. This stage now also includes a small optimisation to take advantage of the quicker calculation of squared distances by adding together adjacent sums of the series of odd numbers (akin to the method used during stage one). Pseudo-code for this new final stage is given in algorithm 7.

6.2.4 Unimplemented Modifications

In addition to the presented extensions, two other modifications were developed. While reducing the algorithmic complexity (the overall number of pixel iterations required to compute the final distance map), the reductions in iterations due to these modifications were not large enough to offset the increased computational overheads required to produce them. The modifications resulted in no significant gains in sequential CPU run-time performance, and in the case of the second, more involved modification, a small performance decrease was observed. As a result, these modifications are left unimplemented in the final algorithm.

The first of these unimplemented modification introduces an early stack reset check by making an initial comparison against the bottom of the candidate nearest points stack (*i.e.* for the first point having an intersection location to the right of the current array index i). Early tests showed that the rate of stack resets enabled by this extra check did not have sufficient impact to warrant the extra computation of t . This early stack reset check would likely be more effective if sufficient measures were not already being used to minimise the size of the candidate points stack.

The second, more involved modification, takes advantage of contiguous row values referring to foreground points at the same orthogonal distance from the current row. This situation is shown in figure 6-9(a).

In figure 6-9(a), the blue foreground points are all at the same height and it is clear that the distances from each of these points to the $y = 0$ intersection with the horizontal axis (the row array) are all minimised by those points. None of the red points can be closer at any location

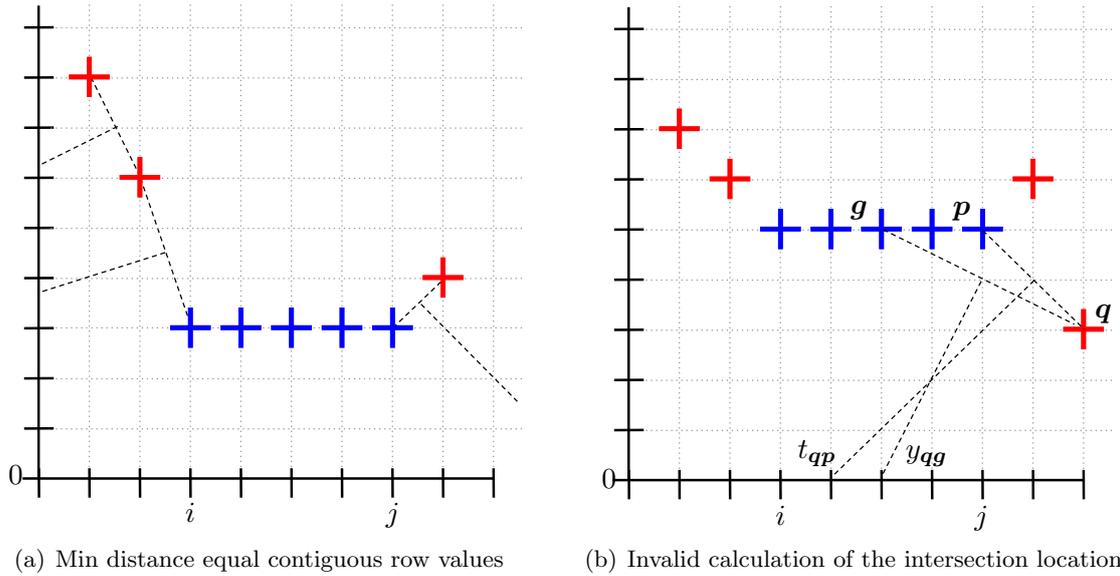


Figure 6-9: In 6-9(a), contiguous regions of equal row value can be ignored to avoid unnecessary updates (only endpoint locations need be stored). In 6-9(b), the presence of a point lower than the contiguous row values complicates the calculation of the intersection location.

along row $y = 0$ within $[i, j]$. This means that the corresponding values at array locations $[i, j]$ can be ignored in the stage three update. However, while endpoints of contiguous regions in the array having values ≤ 1 are always closer to locations outside the region that are adjacent to the endpoints in the array, the same is not true for the endpoints of contiguous regions having arbitrary height. In figure 6-9(b), point q to the right of the row height contiguous region is found to be lower.

This requires calculating the equidistant location of intersection of point q with the correct point *within* the set of contiguous blue foreground points. It is no longer the case that the minimum distance is already set over the whole of the range $[i, j]$ and that updates over the whole of this region can be ignored. If only the endpoints of the blue region are stored, this new intersection location cannot be calculated using equation 6.15. The location of t_{qp} incorrectly determines that row array locations $\geq t_{qp}$ are closer to q than to any of the blue points. To determine the correct intersection location in these cases, the real Euclidean distance must be calculated requiring a square root operation (giving the intersection location in figure 6-9(b) of y_{qg}). This adjustment must be included as part of the main candidate nearest point stack update. This significantly increases computational overheads for little to no practical performance advantage, and a reduction in performance in many cases.

Algorithm 4 EDT Stage 2: Per row identification of candidate nearest points

Ensure: $\text{LENGTH}(npList) \geq \text{LENGTH}(array)$

- 1: **procedure** IDENTIFYNEARESTPOINTS($npList, array$)
- 2: $n \leftarrow \text{LENGTH}(array)$
- 3: $i, minj \leftarrow \text{IGNOREFROMLEFT}(array)$
- 4: **if** $i < n$ **then**
- 5: $npList[0] \leftarrow \{i, array[i], minj, i^2 + array[i]\}$
- 6: $i \leftarrow i + 1$
- 7: **end if**
- 8: $p \leftarrow 0$
- 9: $plast \leftarrow 0$
- 10: $maxj \leftarrow n - 1$ ▷ Ignore values ≤ 1 from the right
- 11: **while** $maxj > i \wedge array[maxj] \leq 1$ **do**
- 12: $maxj \leftarrow maxj - 1$
- 13: **end while**
- 14: $maxj \leftarrow \max(n - 1, maxj + 1)$
- 15: **while** $i \leq maxj$ **do**
- 16: **if** $plast > p \wedge i > npList[p + 1]_b$ **then** ▷ Update closest current point at i
- 17: $p \leftarrow p + 1$
- 18: **end if**
- 19: **if** $array[i] < \infty$ **then**
- 20: $q \leftarrow \{i, array[i], minj, i^2 + array[i]\}$
- 21: **if** $array[i] < 2$ **then** ▷ Store endpoints of regions ≤ 1
- 22: **if** $array[i - 1] > 1$ **then** ▷ Left endpoint
- 23: $plast \leftarrow p$
- 24: UPDATENEARESTPOINTSTACK($array, minj, maxj, npList, p, plast, q$)
- 25: **else if** $array[i + 1] > 1$ **then** ▷ Right endpoint
- 26: $q_b \leftarrow npList[p]_x - 1$
- 27: $plast \leftarrow plast + 1$
- 28: $p \leftarrow plast$
- 29: $npList[p] \leftarrow q$
- 30: **end if**
- 31: **else**
- 32: UPDATENEARESTPOINTSTACK($array, minj, maxj, npList, p, plast, q$)
- 33: **end if**
- 34: **end if**
- 35: $i \leftarrow i + 1$
- 36: **end while**
- 37: **if** $\text{LENGTH}(npList) > 0$ **then**
- 38: **if** $array[maxj] \leq 1$ **then**
- 39: $maxj \leftarrow maxj - 1$
- 40: **end if**
- 41: CALCFINALSQUARED EUCLIDEAN DISTANCE($npList, array, maxj$)
- 42: **end if**
- 43: **end procedure**

Algorithm 5 EDT Stage 2a: Skip over initial points at left

```
1: function IGNOREFROMLEFT(array)
2:    $n \leftarrow \text{LENGTH}(\textit{array})$ 
3:    $i \leftarrow 0$ 
4:   while  $i < n \wedge \textit{array}[i] \geq \infty$  do ▷ Skip points at infinity
5:      $i \leftarrow i + 1$ 
6:   end while
7:    $\textit{minj} \leftarrow -1$ 
8:   if  $i = 0$  then ▷ Only skip foreground points if no prior points at infinity
9:     while  $i < n \wedge \textit{array}[i] < 2$  do
10:       $i \leftarrow i + 1$ 
11:    end while
12:     $\textit{minj} \leftarrow i - 1$ 
13:    if  $i < n$  then ▷ Ensure  $i$  starts at first foreground point
14:       $i \leftarrow \max(0, i - 1)$ 
15:    end if
16:  end if
17:  return  $i, \textit{minj}$ 
18: end function
```

Algorithm 6 EDT Stage 2b: Update the stack of candidate nearest points with new point q

```
1: procedure UPDATENEARESTPOINTSTACK(array,  $\textit{minj}$ ,  $\textit{maxj}$ , npList,  $\textit{pidx}$ ,  $\textit{plast}$ ,  $q$ )
2:    $p \leftarrow \textit{npList}[\textit{plast}]$ 
3:    $t \leftarrow \text{CALCT}(q_x, q_h, p_x, p_h)$ 
4:   while  $t \leq p_b \wedge \textit{plast} > 0$  do
5:      $\textit{plast} \leftarrow \textit{plast} - 1$ 
6:      $p \leftarrow \textit{npList}[\textit{plast}]$ 
7:      $t \leftarrow \text{CALCT}(q_x, q_h, p_x, p_h)$ 
8:   end while
9:   if  $t < 0$  then
10:     $\textit{npList}[0] \leftarrow q$ 
11:     $q_b \leftarrow \textit{minj}$ 
12:     $\textit{pidx} \leftarrow 0$ 
13:   else if  $(t > \textit{maxj}) \vee (t \geq q_x \wedge \textit{array}[t + 1] \leq \text{SQDIST}(q, t + 1))$  then
14:     return
15:   else
16:      $q_b \leftarrow t$ 
17:      $\textit{plast} \leftarrow \textit{plast} + 1$ 
18:      $\textit{npList}[\textit{plast}] \leftarrow q$ 
19:     if  $\textit{plast} < \textit{pidx}$  then
20:        $\textit{pidx} \leftarrow \textit{plast}$ 
21:     end if
22:   end if
23: end procedure
```

Algorithm 7 EDT Stage 3: Squared Euclidean distance calculation over non-foreground

```
1: procedure CALCFINALSQUAREDEUCLIDEANDISTANCE(npList, array, rowEndIdx)
2:    $j \leftarrow \text{rowEndIdx}$ 
3:   for  $i \in [\text{LENGTH}(\text{npList}) - 1, 0]$  do
4:      $np \leftarrow \text{npList}[i]$ 
5:      $dsum \leftarrow np_d + (j - np_x)^2$ 
6:      $d \leftarrow 2(j - np_x) - 1$ 
7:     while  $j > \max(np_x, np_b)$  do ▷ Update up to right most bound
8:        $\text{array}[j] \leftarrow dsum$ 
9:        $dsum \leftarrow dsum - d$ 
10:       $d \leftarrow d - 2$ 
11:       $j \leftarrow j - 1$ 
12:    end while
13:     $nq \leftarrow \text{npList}[i - 1]$ 
14:    if  $np_d < 2 \wedge nq_d < 2 \wedge np_b < nq_x$  then ▷ Skip contiguous foreground
15:       $j \leftarrow np_b$  ▷ Left exclusive bound less than next nearest point's location
16:    else
17:      while  $j > np_b$  do ▷ Point not part of contiguous region of foreground
18:         $\text{array}[j] \leftarrow dsum$ 
19:         $dsum \leftarrow dsum - d$ 
20:         $d \leftarrow d - 2$ 
21:         $j \leftarrow j - 1$ 
22:      end while
23:    end if
24:  end for
25: end procedure
```

6.3 Experimental Methodology

Algorithms for computing the EDT can be compared in terms of their time and space requirements, their ability to exactly compute the EDT, their simplicity (or ease of understanding/implementation) and their extensibility to different domains. Of these aspects, arguably the most important issue for practical purposes is the speed of calculation. While space requirements are still a consideration (especially for some of the older algorithms tested *e.g.* Saito and Toriwaki (1994); Cuisenaire and Macq (1999); Lotufo and Zampirolli (2001)), none of the more recent algorithms have especially severe memory requirements. In particular, the algorithmic framework as described in section 6.2 (as used in the algorithms of Meijster *et al.* (2000), Maurer *et al.* (2003), Felzenszwalb and Huttenlocher (2004) and Wang and Tan (2013)) has memory requirements proportional to the square root of the size of the image – not including memory required to store the image itself.

Previous studies have compared the performance of different algorithms according to their real-time CPU sequential execution speeds. Variability in implementation characteristics (language, compiler/optimisation, platform *etc.*) can adversely affect the reliability of such results. In these previous studies, all algorithms were compiled within the same language, and using the same optimisation flags. However, it is still the case that in real world use, manual optimisations, or choosing to implement a particular algorithm in a well suited language, or understanding that a given algorithm works better under particular conditions with well controlled input data, can still throw doubt onto scientific conclusions that any single algorithm offers superior performance in the general case.

By itself, the analysis of real-time CPU performance (though of ultimate practical importance) does not provide a reliable enough measure of the true computational complexity of an algorithm. If combined with an understanding of the dynamic computational complexity of an algorithm, more reliable conclusions can be made concerning the behaviour of an algorithm under a range of inputs, especially with regard to the stability of any particular algorithm for datasets of different character. Computational complexity is independent of implementation or platform, hence it can be used as a more impartial metric to compare the effectiveness of different algorithms, and the analysis remains valid in the face of continued technological improvements.

For any particular EDT algorithm, code that appears long and complex statically can perform very well dynamically on real inputs (explaining the practical usefulness of algorithms that have super-linear static computational complexity). The nature of the calculation of the EDT means that it is difficult to empirically evaluate the actual number of instructions

performed by each algorithm on any given input image. The logic of the various EDT algorithms is generally dependent on factors other than the overall size of the input such as image aspect ratio, the ratio of foreground to background points, the homogeneity of foreground/background regions, the distribution of the foreground/background points, and the morphology of the foreground/background regions (for example, the degree to which straight edges are represented or if the image contains convex foreground objects). These factors can affect the algorithms in different ways, and so it is necessary to test the algorithms against a range of input data that are diverse enough to address such factors so that evaluation does not unfairly bias any particular algorithm that might take advantage of one or more of these factors.

In this section, the dynamic computational complexity and the real-time sequential CPU performance of the different algorithms are empirically evaluated. Most of the algorithms tested have linear worst case complexity, though in some cases, the constant of complexity can be quite high.

Counting the actual number of operations carried out by each algorithm as they work over different input data is generally not practicable so a proxy measure for the dynamic computational complexity of an algorithm is used. This is achieved by embedding loop counters in the source code of each of the algorithms and by taking note of long operations (such as array copying). Although these counts do not consider the constant computation factor for a given iteration of a loop (*i.e.* the number of instructions carried out in the body of a loop), the counts can provide a good estimate of the computational complexity of the algorithm as a linear factor of the actual computational complexity on a given input. Since the biggest factor in the actual computational complexity is the number of times memory (image locations) must be referenced, counting loop iterations is a sensible indicator of true algorithmic performance; finding the algorithms with the fewest pixel iterations / loop counts gives a reasonable indication of which algorithms offer the fastest actual algorithmic implementation over a range of inputs.

Even though loop counters are instructive in comparing how the different algorithms vary in their behaviour over different input data, it is not necessarily warranted to compare this statistic directly between the different algorithms because the complexity of statement blocks can vary a great deal across algorithms. Loop iteration counts are provided to help ascertain the stability of each of the algorithms over a range of inputs.

Of more practical relevance are results giving the amount of CPU time required by each of the algorithms. The cost of incrementing the pixel iteration counters is an added cost to the run-time of each algorithm, but this cost is negligible and is roughly equal across all of the

algorithms tested so it can be ignored for the purposes of comparison.

The algorithms selected for evaluation are the same as tested by Wang and Tan (2013) with the addition of the new algorithm using the extensions as presented in this chapter. In addition, even though it does not give the exact distance transform, the Chamfer 3-4 algorithm (Barrow *et al.*, 1977; Borgefors, 1988) is included to give a baseline comparison of the performance of each algorithm. In many applications where exact distance calculations are less important, the Chamfer 3-4 algorithm (or algorithms of a similar nature) are used because they are generally faster and give more stable performance than exact Euclidean distance transform algorithms. The algorithms are listed in table 6.1 together with their type and worst case asymptotic complexity.

Method	Short Name	Complexity	Type
Palmer <i>et al.</i> (2014)	RP	$O(N)$	Ind. Scanning
Felzenszwalb and Huttenlocher (2004)	PFF	$O(N)$	Ind. Scanning
Wang and Tan (2013)	PBEDT	$O(N)$	Ind. Scanning
Lotufo and Zampirolli (2001)	LZ	$O(N^{\frac{3}{2}})$	Ind. Scanning
Maurer <i>et al.</i> (2003)	MAURER	$O(N)$	Ind. Scanning
Meijster <i>et al.</i> (2000)	MEIJ	$O(N)$	Ind. Scanning
Cuisenaire and Macq (1999)	CUIS	$O(N)$	Ord. Propagation
Saito and Toriwaki (1994)	SAITO	$O(N^{\frac{3}{2}})$	Ind. Scanning
Chamfer 3-4 (<i>not Euclidean</i>)	CHAM34	$O(N)$	Raster Scanning

Table 6.1: Distance Transform Algorithms Evaluated

The algorithms are empirically evaluated over several different datasets. Each image from each dataset is provided as binary input to each of the algorithms in turn and the results collected and recorded. In the case of non-binary images, the images are contrast stretched and thresholded at each of the threshold levels before being inverted and thresholded again at each of the levels. Contrast stretching helps ensure that the threshold levels are applicable to every input image and give good distributions of foreground and background regions.

6.3.1 Datasets

In previous work, these algorithms have been evaluated primarily against artificial datasets (Fabri *et al.*, 2008; Wang and Tan, 2013). No thorough evaluation on “natural” real world imagery has been conducted. The computational complexity (and the correlated CPU run-time performance) of all of these EDT algorithms is highly dependent on the nature of the image and so it is important to test using a broad selection of different data so as to not unfairly favour

any particular algorithm if one is to make a general conclusion about practical performance. In this work, five different datasets are used. These are summarised in table 6.2.

Dataset	Image Type	Data Points	Image Dimensions
Fabbri <i>et al.</i> (2008)	Artificial (Binary)	114	3000×3000 and 4000×4000
MARS	Natural (Colour)	555	7000×1764 to 7000×9419
MARS EDGE	Natural (Binary)	37	7000×1764 to 7000×9419
NATURAL	Natural (Colour)	240	7000×4650 to 7000×9333
RANDOM LINES	Artificial (Binary)	50	4057×5779 and 5779×4057

Table 6.2: Datasets used for evaluation of Euclidean Distance Transform algorithms

Fabbri *et al.* (2008) Dataset

The first dataset consists of the larger images (3000×3000 and 4000×4000 pixel dimension) from the dataset used in the study of Fabbri *et al.* (2008). CPU processing power has increased substantially, hence only the larger images are used to give more discriminative results across the different algorithms. These images are all binary and consist of 11 images of random dots covering varying proportions of the image. The coverage proportions are 1, 10, 20, 30, 40, 50, 60, 70, 80, 90 and 99 percent. Both 3000^2 and 4000^2 pixel images are available for these examples giving 22 images. Also within this dataset are images of squares of different sizes and rotation angles, all again with different coverage proportions. These images are only given in 3000^2 pixel dimensions. The coverage proportions (of the squares) in these images are 15, 30, 50, 70, and 95 percent. Seven images for each coverage proportion give squares in different angles of rotations: 0° , 30° , 45° , 60° , 75° , 90° (the 0° and 90° images are duplicates). This gives 35 images of squares and 57 total images in this dataset. Finally, to increase the robustness of the results, all of the images in this dataset are negated swapping “background” with “foreground” pixels and incorporated into the original dataset giving a final total of 114 images. Two examples from this dataset are given in figure 6-10.

MARS Dataset

The second test dataset comprises a selection of 37 images of the landscape of the planet Mars as imaged by various rovers and orbiting spacecraft courtesy of NASA/JPL-Caltech (2014). These are all colour images of varying original large image dimensions scaled up to a common width of 7000 pixels wide and a varying number of pixels in height ranging from 1764 to 9419. Some of the images also contain artificial components such as photographic

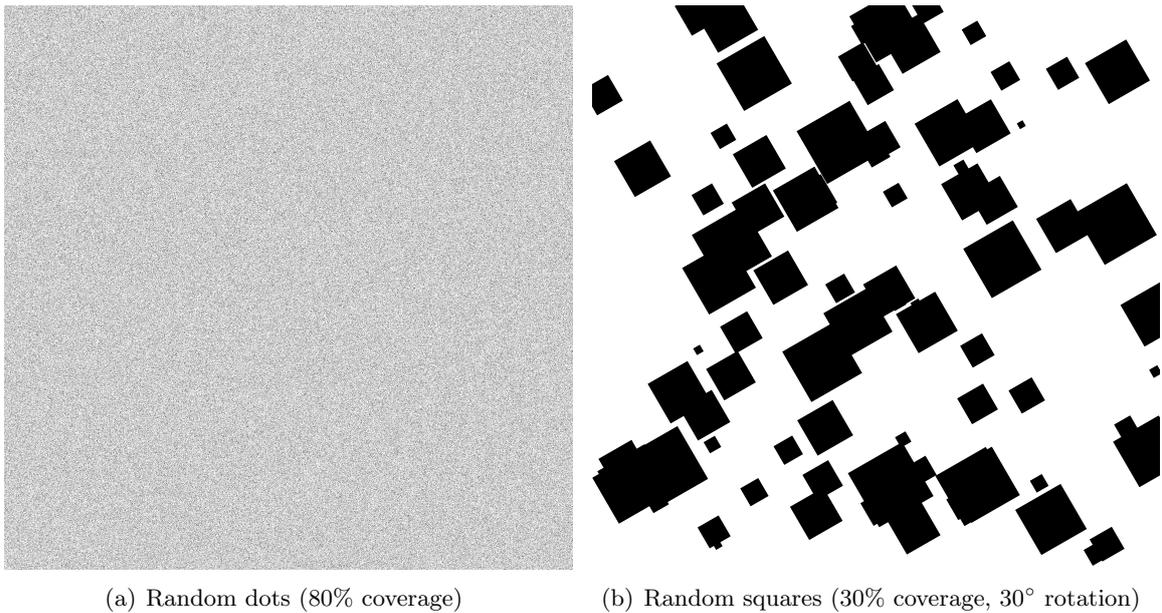


Figure 6-10: Sample test images from the Fabbri *et al.* (2008) dataset

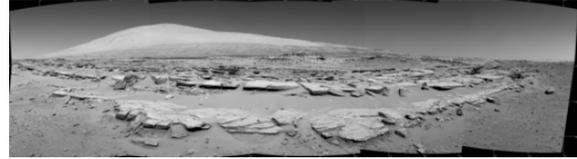
measurements and other notations. The varying aspect ratios of the images helps to avoid any bias towards algorithms that perform better on square images. The images give a range of interesting non-random naturally occurring features from a specific domain. For testing, because these images are not binary, seven different threshold levels are used to generate images with “foreground” and “background” components. The threshold levels used are 36, 72, 108, 144, 180, 216, and 252. The images are also inverted in the RGB colour space and the same seven threshold values used for the inverted images. Additionally, a threshold level of zero is used to produce images of all “foreground” or all “background” to test the edge case performance of the algorithms. Four examples from this dataset are shown in figure 6-11.

MARS EDGE Dataset

The third dataset uses the same imagery as the second dataset, but instead of applying multiple thresholds, the images are processed to generate edge images using the Canny edge detector (Canny, 1986) (using parameters of 10 and 120 for the hysteresis first and second thresholds respectively). These edge images are then treated as binary images giving 37 data points.



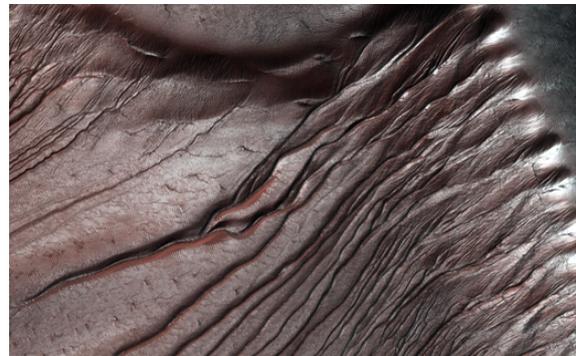
(a) The Curiosity rover's view of a Martian dune after crossing it



(b) Martian landscape with rock rows and mount sharp imaged by the Curiosity rover



(c) Martian dunes imaged from the Mars Reconnaissance Orbiter



(d) Autumn frost accumulation on dunes as imaged from the Mars Reconnaissance Orbiter

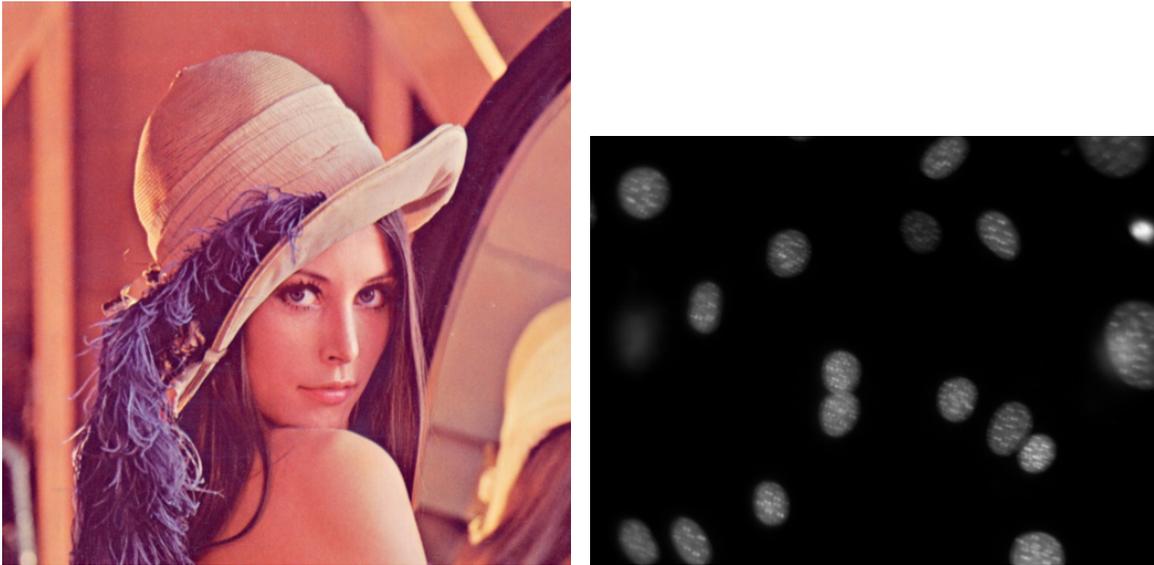
Figure 6-11: Sample images from the Mars dataset courtesy of NASA/JPL-Caltech (2014)

NATURAL Dataset

The fourth dataset is comprised of a selection of 16 widely varying non-artificial, colour (or grey-scale) images obtained from a variety of public domain sources. These images are also inverted and scaled to provide a fixed width of 7000 pixels and a varying height (images have different aspect ratios). The same threshold values as for the Mars dataset are used giving 240 total test cases for each algorithm. Figure 6-12 gives two examples from this dataset.

RANDOM LINES Dataset

The final dataset is comprised of 50 artificially generated images of randomly located and oriented lines of varying length and density. Image dimensions (width \times height) are either



(a) Lena Söderberg

(b) Cell nuclei (from the Python Image Tutorial)

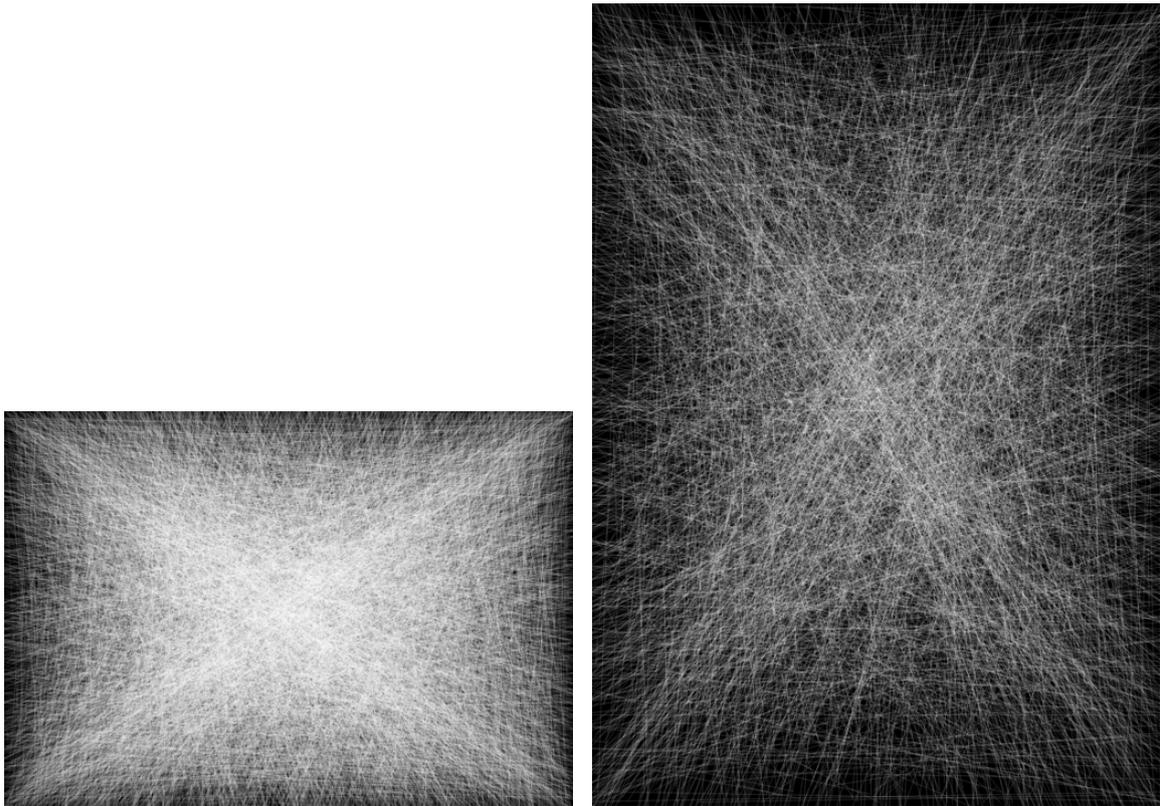
Figure 6-12: Sample test images from the Natural (Misc) dataset

4057×5779 or 5779×4057 . Prime dimensions are used to test algorithms that might take advantage of images that are easily subdivided into regions of equal size. The number of lines generated in each image is fixed (from 1 to 20000), but since their location, length, and angle of orientation is randomly selected, the total coverage of the lines in an image increases non-uniformly from $\approx 0\%$ to 84% . Figure 6-13 shows two sample images from this dataset.

6.3.2 Exactness

In previous evaluations, all of the algorithms tested were empirically found to produce the exact Euclidean distance maps (by comparison against brute force exemplars) (Fabbri *et al.*, 2008; Wang and Tan, 2013). In this work, it was found that the algorithm by Maurer *et al.* (2003) as implemented by Fabbri *et al.* (2008) (and as used in those previous studies) does not produce the exact distance map for one of the test images used in the study of Fabbri *et al.* (2008). The erroneous result is shown in figure 6-14.

This empirical error was not addressed in the study by Fabbri *et al.* (2008). Because the algorithm's exactness is theoretically proven (Maurer *et al.*, 2003), it is assumed that the actual implementation is in error, rather than the theoretical algorithm *per se*. As such, all results for this EDT algorithm are retained, with the caveat that fixing the implementation



(a) 7981 random lines giving 59% foreground coverage (b) 2794 random lines giving 28% foreground coverage

Figure 6-13: Sample test images from the Random Lines dataset

should not affect the resulting performance statistics too greatly.

In several cases, an inexact distance map is also produced by the original implementation of the algorithm given by Felzenszwalb and Huttenlocher (2004). This algorithm is similar in fundamental respects to the other algorithms evaluated, hence it is again presumed that fixing the implementation should not significantly impact upon the resulting run-time performance of the algorithm. In the case of this algorithm, inexact EDTs are only generated for very high resolution images (*i.e.* more than a few thousand pixels on either edge). On images having fewer pixels the algorithm typically produces the exact EDT. It is noted that this algorithm, unlike the method of Wang and Tan (2013), uses a real valued calculation of the parabola intersection rather than integer division and that the nature of the errors in very large images suggests that propagation of inexact representations of floating point values may be to blame.

Implementations of all of the algorithms except PBEDT, PFF and RP are provided as part of the Animal computer vision library (Fabbri, 2013). Source code for PBEDT is provided

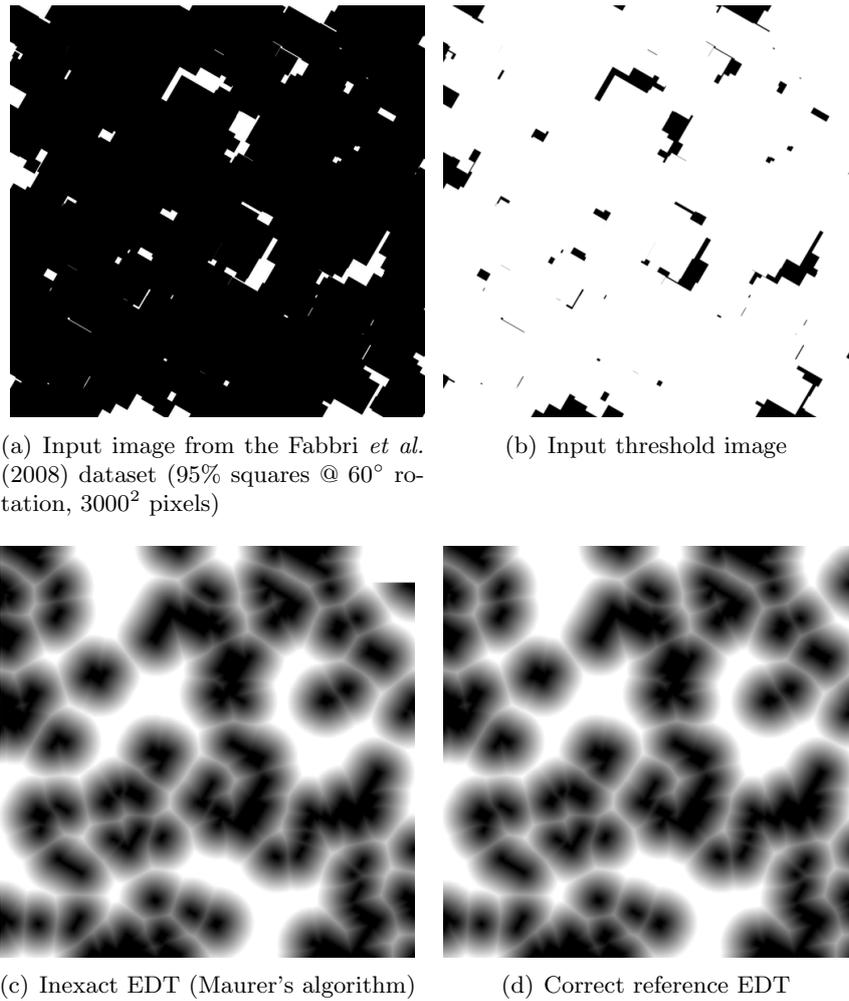


Figure 6-14: Inexact calculation by an implementation of Maurer *et al.* (2003)'s distance transform algorithm Fabbri *et al.* (2008) (see the top right corner of figure 6-14(c))

by the authors (Wang and Tan, 2013) and is compiled into the source code of the Animal computer vision library. Source code for PFF is provided by the authors (Felzenszwalb and Huttenlocher, 2004) and is separately compiled along with the implementation of the new (RP) algorithm. All algorithms are implemented as C/C++ and compiled as C++ using the Free Software Foundation's g++ compiler version 4.8.2¹. To ensure parity of machine code, optimisation flags for compilation of the Animal computer vision library and the PFF and RP algorithms and all associated test harnesses were set equal to level three optimisation (-O3). All testing was single threaded and performed on an Intel Core i7-3632QM processor with 8GBs RAM. Portable CPU timing mechanisms were provided using the Boost² C++ library.

¹<https://gcc.gnu.org/>

²<http://www.boost.org/>

6.4 Results

Results for the Fabbri *et al.* (2008) dataset giving the mean, max and min CPU processing times (in seconds) along with the mean number of iterations required to produce the distance transforms are given in table 6.3. Results corresponding to the MARS, MARS EDGE, NATURAL and RANDOM LINES datasets are shown in tables 6.4, 6.5, 6.6, and 6.7 respectively.

Algorithm	Mean CPU	Max CPU	Min CPU	Mean Iterations
RP	0.15	0.48	0.04	2.70
PFF	0.23	0.47	0.13	3.33
PBEDT	0.21	0.48	0.14	3.49
LZ	0.24	1.22	0.06	12.16
MAURER	0.21	0.50	0.13	4.74
MEIJ	0.23	0.65	0.14	5.08
CUIS	0.27	1.05	0.06	3.81
SAITO	0.17	0.51	0.05	8.78
CHAM34	0.11	0.18	0.09	2.00

Table 6.3: Fabbri *et al.* (2008) dataset results

Algorithm	Mean CPU	Max CPU	Min CPU	Mean Iterations
RP	0.51	2.13	0.05	2.65
PFF	0.87	2.60	0.17	3.23
PBEDT	0.73	2.29	0.10	3.37
LZ	1.37	19.18	0.08	13.54
MAURER	0.70	2.10	0.10	4.62
MEIJ	0.78	2.19	0.16	5.07
CUIS	1.27	9.58	0.09	4.39
SAITO	0.87	12.04	0.06	11.52
CHAM34	0.55	1.43	0.13	2.00

Table 6.4: MARS dataset results

Algorithm	Mean CPU	Max CPU	Min CPU	Mean Iterations
RP	0.75	2.13	0.05	3.22
PFF	1.02	2.60	0.23	3.47
PBEDT	0.77	2.13	0.06	3.75
LZ	1.88	21.59	0.12	15.40
MAURER	0.79	2.06	0.10	4.29
MELJ	0.90	2.25	0.25	5.13
CUIS	2.15	7.82	0.08	4.97
SAITO	1.01	2.71	0.10	11.23
CHAM34	0.41	1.05	0.11	2.00

Table 6.5: MARS EDGE dataset results

Algorithm	Mean CPU	Max CPU	Min CPU	Mean Iterations
RP	0.50	1.22	0.18	2.67
PFF	0.90	1.65	0.57	3.23
PBEDT	0.77	1.63	0.26	3.37
LZ	1.42	6.97	0.28	18.64
MAURER	0.71	1.34	0.33	4.60
MELJ	0.80	1.34	0.55	5.09
CUIS	1.13	3.27	0.20	4.10
SAITO	1.07	6.68	0.19	15.79
CHAM34	0.44	0.68	0.32	2.00

Table 6.6: NATURAL dataset results

Algorithm	Mean CPU	Max CPU	Min CPU	Mean Iterations
RP	0.38	0.62	0.16	2.41
PFF	0.58	0.68	0.49	3.23
PBEDT	0.54	0.59	0.21	3.29
LZ	0.48	2.86	0.31	6.91
MAURER	0.63	0.97	0.29	4.86
MELJ	0.64	0.82	0.47	5.04
CUIS	0.65	1.51	0.33	3.72
SAITO	0.34	1.99	0.21	6.36
CHAM34	0.24	0.25	0.23	2.00

Table 6.7: RANDOM LINES dataset results

Figure 6-16 shows box-plots for the median and interquartile ranges of the CPU run-times (in seconds) and the median and interquartile ranges for the pixel iterations required to produce the distance transform images from the Fabbri *et al.* (2008) dataset. Figures 6-17, 6-18, 6-19, and 6-20 give the same information for the MARS, MARS EDGE, NATURAL, and RANDOM LINES datasets respectively.

For every dataset apart from the RANDOM LINES dataset, the new algorithm (RP) gives a faster mean CPU processing time than any of the other Euclidean distance transform algorithms. In the RANDOM LINES dataset, in all cases except one, SAITO is faster than RP by ≈ 0.04 seconds, but in a single case the processing time of SAITO is more than ten times slower than RP (1.99 vs 0.16 seconds). This particular data point was verified multiple times to ensure that this processing outlier was in fact indicative of the speed of SAITO in this case. This particular input image also resulted in two of the slowest processing times for the LZ and CUIS algorithms (2.86 and 1.34 seconds respectively) indicating that this lack of stability on some inputs is shared by algorithms other than SAITO. However, this same input image gave some of the fastest processing times on the other algorithms tested (RP: 0.16, PFF: 0.58, PBEDT: 0.21, MAURER: 0.29, MEIJ: 0.47, CHAM34: 0.24), and the time of 0.16 seconds for RP was the fastest processing time for any test image from this dataset.

Figure 6-20(a) shows the outlier for this input image on a number of algorithms (in the case of the LZ algorithm, the outlier is not shown as it is off the top of the graph). When comparing the behaviour of the new algorithm (RP) to the most recently evaluated algorithm (and the one to which it is most closely related - PBEDT - in figure 6-20(a), the advantages given by the new extensions are clearly evident. RP was only marginally slower than PBEDT in this dataset in only four out of fifty cases.

In the results for the original dataset of Fabbri *et al.* (2008) (large images only), RP is the fastest algorithm with the next closest being SAITO. On this dataset, SAITO is more stable over all of the test inputs than on the RANDOM LINES dataset, and the LZ and CUIS algorithms again show poor stability. For the remainder of the algorithms, the variability in results is not great enough to draw reliable conclusions as to which of the five of RP, PFF, PBEDT, MAURER and MEIJ give better performance overall even though RP appears to show the best performance over most of the inputs. The Chamfer 3-4 algorithm is fastest and most stable overall on this dataset. To conclude from this dataset alone, it would appear that the older SAITO algorithm offers performance comparable to the new algorithm (RP) and better performance than the newer PFF, PBEDT, MAURER and MEIJ algorithms. However, this conclusion does not hold when comparing against the results from the other datasets.

In the MARS dataset, figure 6-17(a) shows a much clearer separation in performance between the older LZ, CUIS and SAITO algorithms, and the newer RP, PFF, PBEDT, MAURER and MEIJ algorithms. This distinction is also evident in the results from the NATURAL dataset (figure 6-19(a)). SAITO is faster than LZ and CUIS but is not the most stable (it has the second slowest performance for any individual input after LZ on both the MARS and NATURAL datasets). Given the results on the MARS and NATURAL datasets, it can be concluded that for general use, if predictable, stable performance is required, the LZ, CUIS and SAITO algorithms should be avoided.

The results from the MARS dataset also provide a clearer indication of the performance differences between the five remaining EDT algorithms. RP gives faster performance overall than any of the EDT algorithms even though the RP, PFF, PBEDT, MAURER and MEIJ algorithms all exhibit similar performance stability. Compared to PBEDT, RP gives performance at least as fast as PBEDT in 454 cases out of 556. On the MARS dataset, only MAURER has a smaller maximum CPU run-time than RP (though this difference is not significant). However, on the particular instance giving that result, the degree of inexactness in the EDT produced by MAURER (as implemented by Fabbri *et al.* (2008)) is quite large compared to the exact distance map produced by RP (see figure 6-15 for a comparison) and so the results for MAURER are not as reliable.

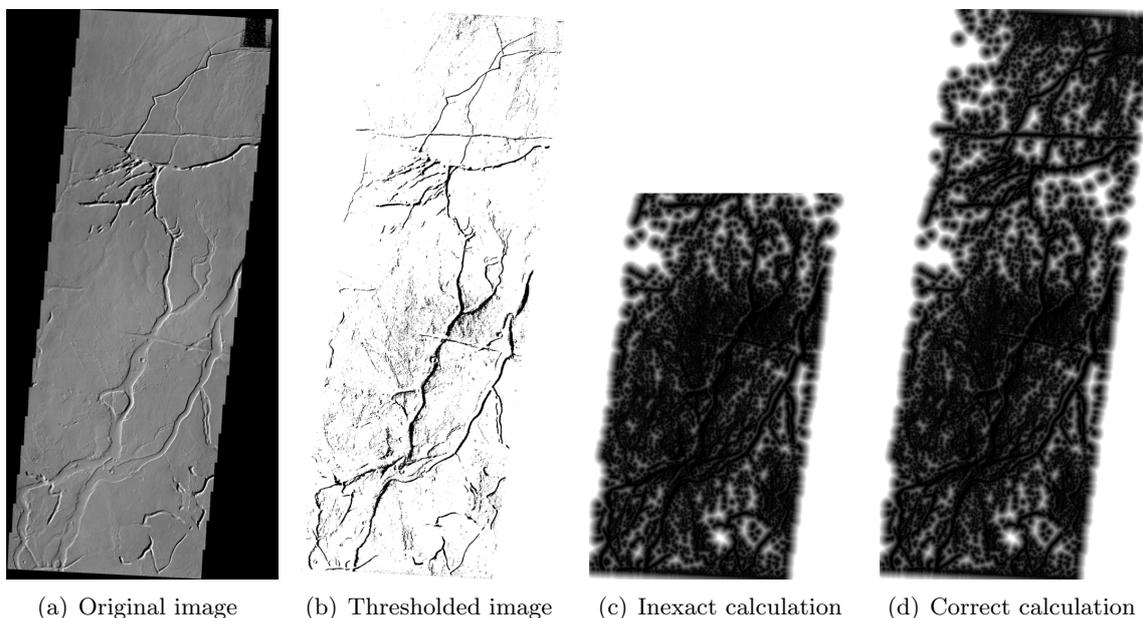


Figure 6-15: Inexact calculation of the EDT by Maurer *et al.* (2003)'s algorithm (Fabbri *et al.*, 2008) on the Mars dataset. Original image courtesy of NASA/JPL-Caltech (2014).

Similar conclusions relating to the relative performance of the different algorithms can be

drawn from the results for the NATURAL dataset (figure 6-19). RP again, gives consistently better performance over all of the other algorithms evaluated.

Finally, on both the MARS and NATURAL datasets, RP is the only algorithm to give faster performance on average than the (inexact) Chamfer 3-4 algorithm (though the Chamfer 3-4 algorithm is much more stable). This is because the RP algorithm is able to ignore large swathes of the image that are already foreground whereas the Chamfer 3-4 algorithm must parse every pixel. These extensions in the RP algorithm have less of an effect in the artificial images because foreground regions are typically more broken up (less homogeneous) and so the average length of “skippable” foreground regions in the artificial images is reduced. As seen in figure 6-18(a), the results from the MARS EDGE dataset support this explanation. The degree of speed up in the new algorithm (RP) is far less in this dataset and overall performance is approximately on parity with PBEDT. In terms of raw CPU processing speed, in the MARS EDGE dataset, RP is no better (though no worse) than the current state-of-the-art PBEDT algorithm. However, as seen in table 6.5 and figure 6-18(b), the RP algorithm is still more efficient than PBEDT in terms of the number of iterations required by the algorithm (3.22 vs 3.75). This is because even though there are fewer “skippable” foreground regions (and those that do occur are of shorter length), the parsing of a foreground point still results in the discard of previously stored candidate nearest points meaning that there are fewer comparisons against previous stored points. However, the relative sparsity of foreground points in these edge maps means that this improvement in efficiency does not translate into an overall speed up in run-time performance; the added complexity of the required algorithmic logic is not being offset by the potential gains of this optimisation in this case. Table 6.5 also shows that the performance for the LZ and CUIS algorithms is typically much worse than the other algorithms on this dataset (though the performance of SAITO is comparable to PFF in these results).

The results on the MARS and NATURAL datasets imply that if non-edge images are being processed, and algorithmic stability is not as important as overall speed (*e.g.* in applications where very many natural images are being processed in rapid succession), the RP algorithm can be a better performing choice than even inexact algorithms such as the Chamfer 3-4 algorithm, not only from the perspective of giving more accurate results, but from the perspective of being faster to compute overall. However, given the results from the Fabbri *et al.* (2008), MARS EDGE and RANDOM datasets, if edge images are being processed, the choice of algorithm depends primarily on accuracy requirements. If accuracy is not as important as speed, inexact algorithms such as the Chamfer 3-4 algorithm should be preferred. If accuracy is important, RP or PBEDT should be used (though with higher proportions of foreground points, RP will give better performance).

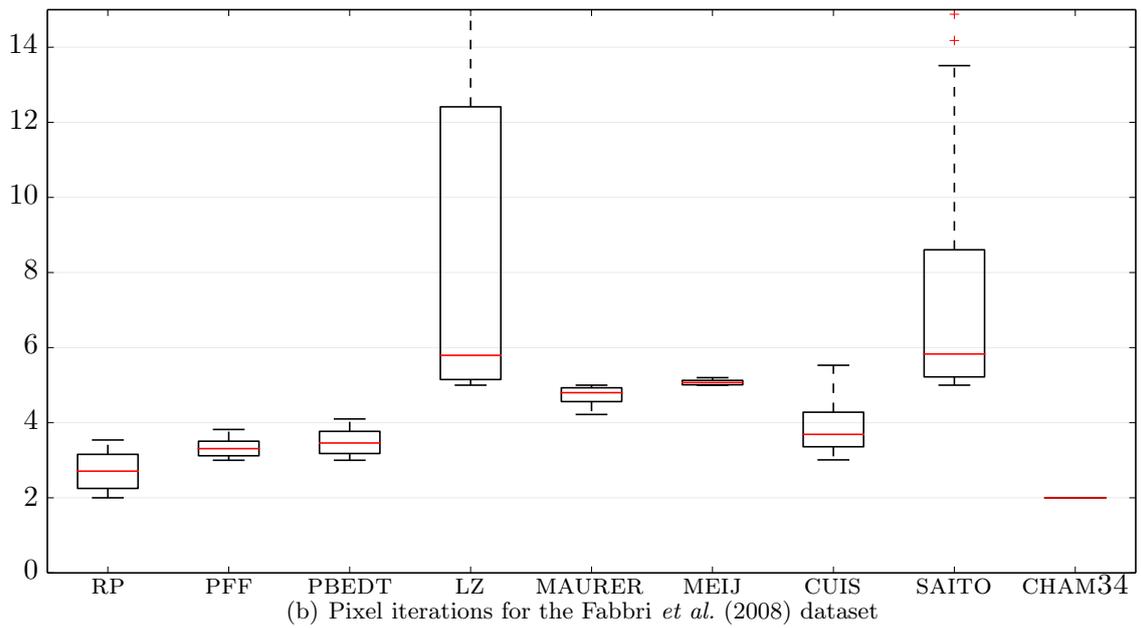
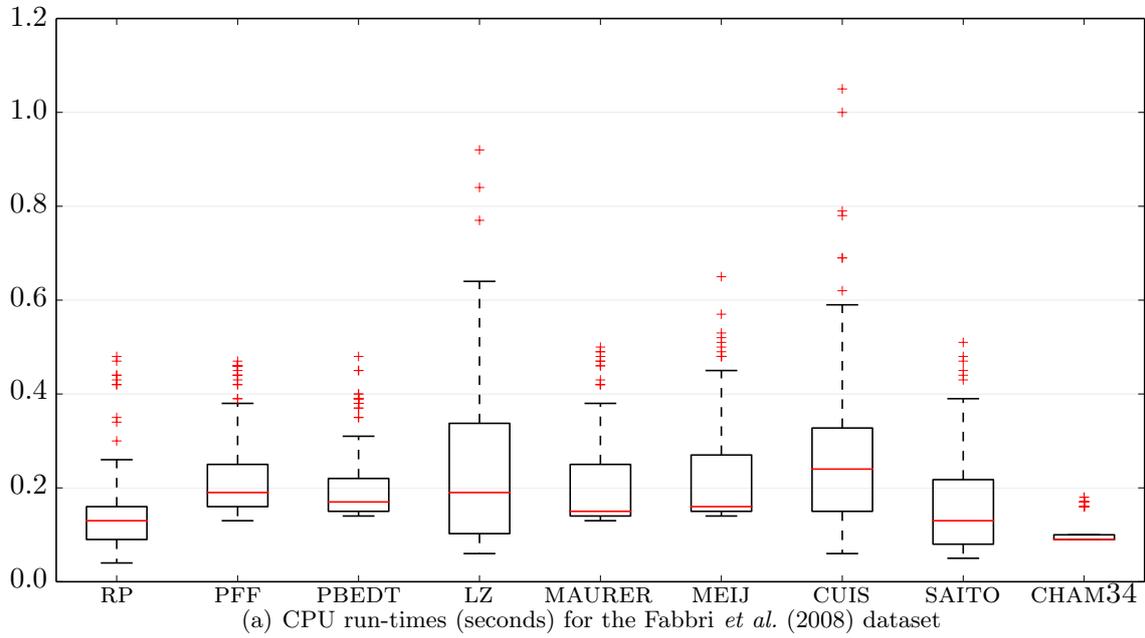


Figure 6-16: Box-plots showing per algorithm median and interquartile range results for the Fabbri *et al.* (2008) dataset

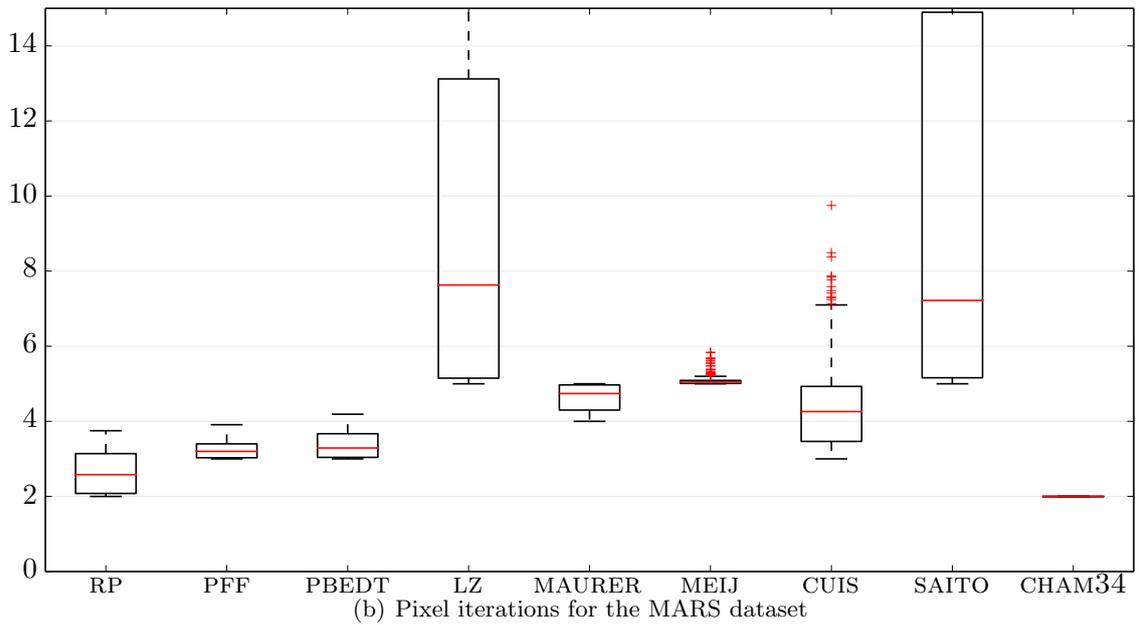
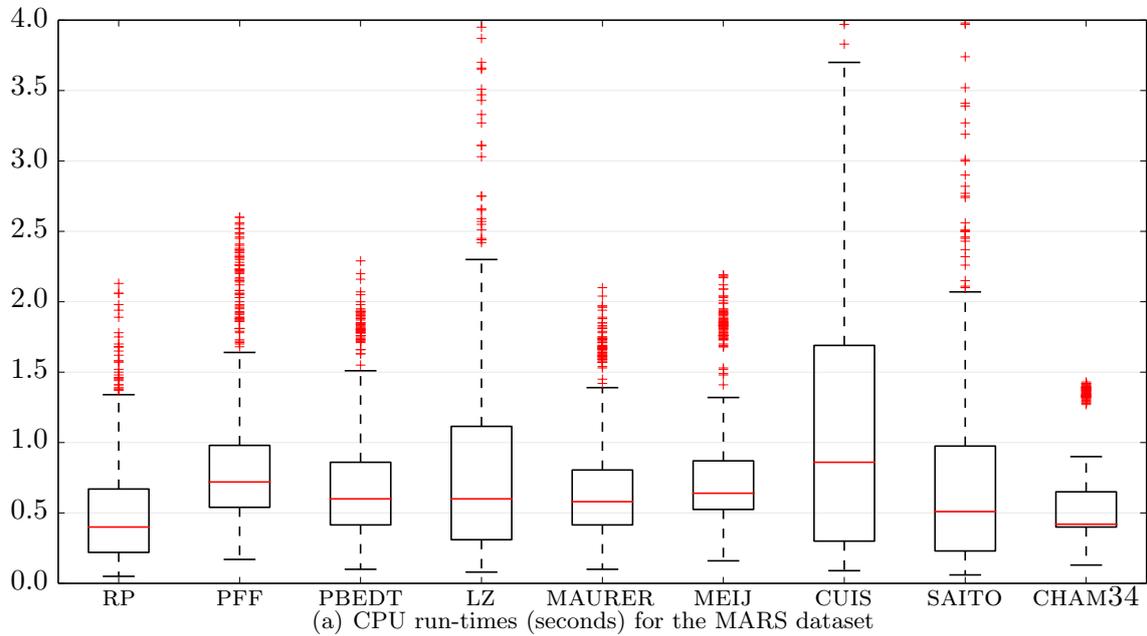


Figure 6-17: Box-plots showing per algorithm median and interquartile range results for the MARS dataset

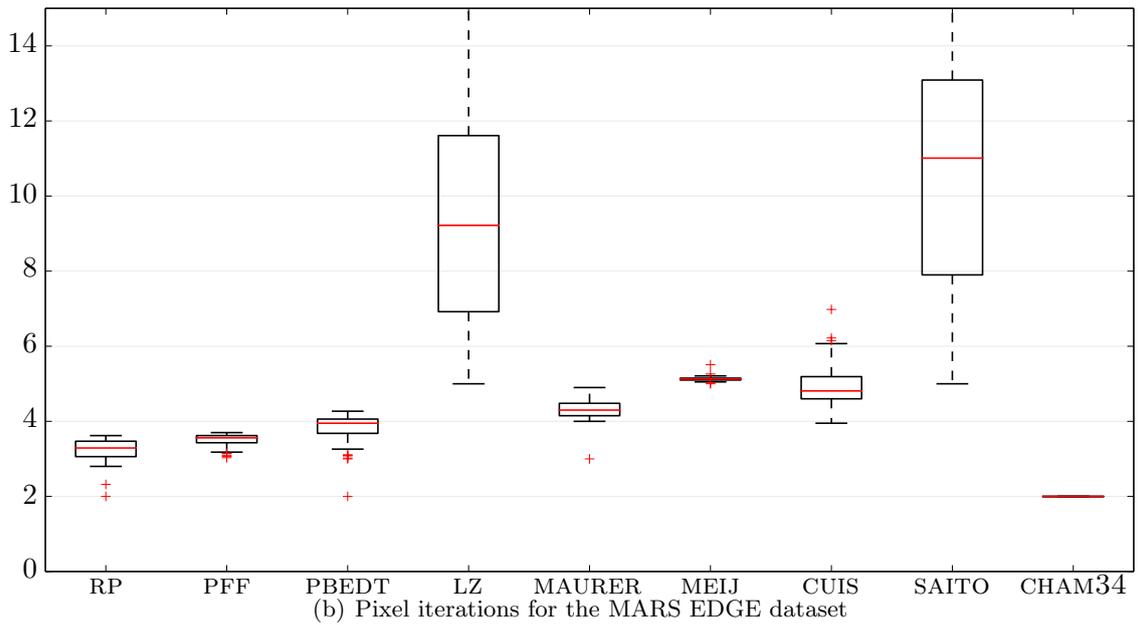
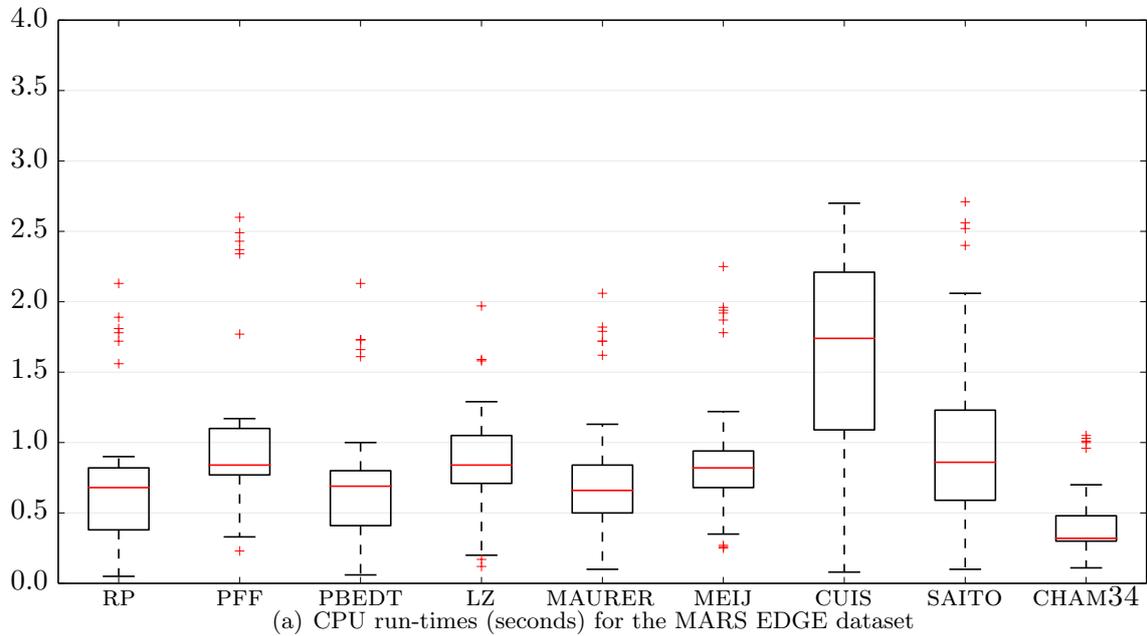


Figure 6-18: Box-plots showing per algorithm median and interquartile range results for the MARS EDGE dataset

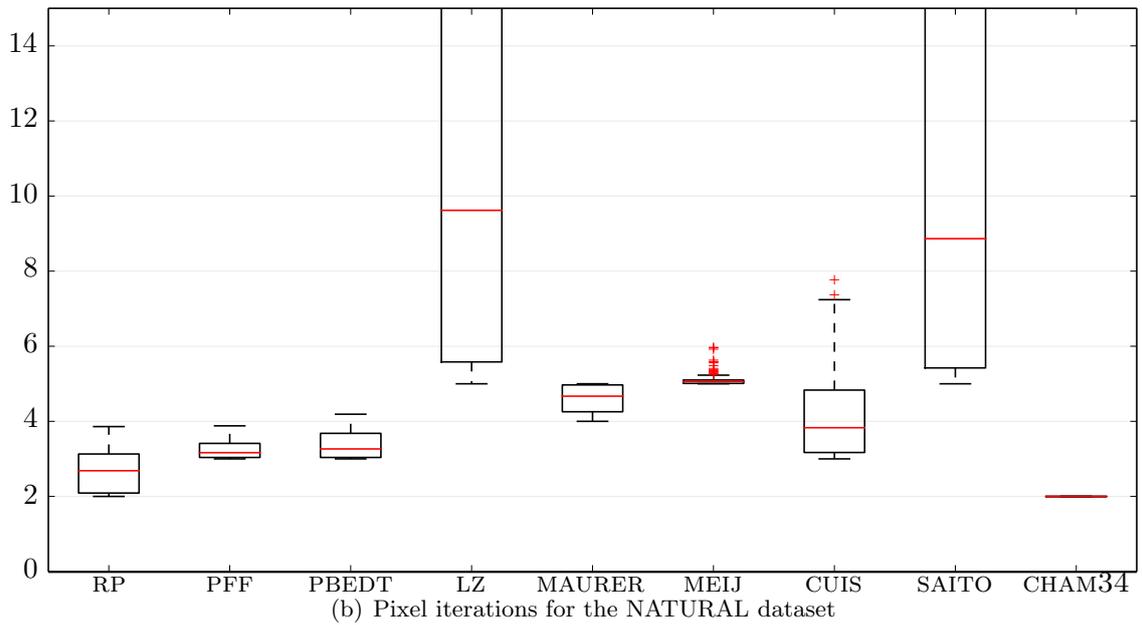
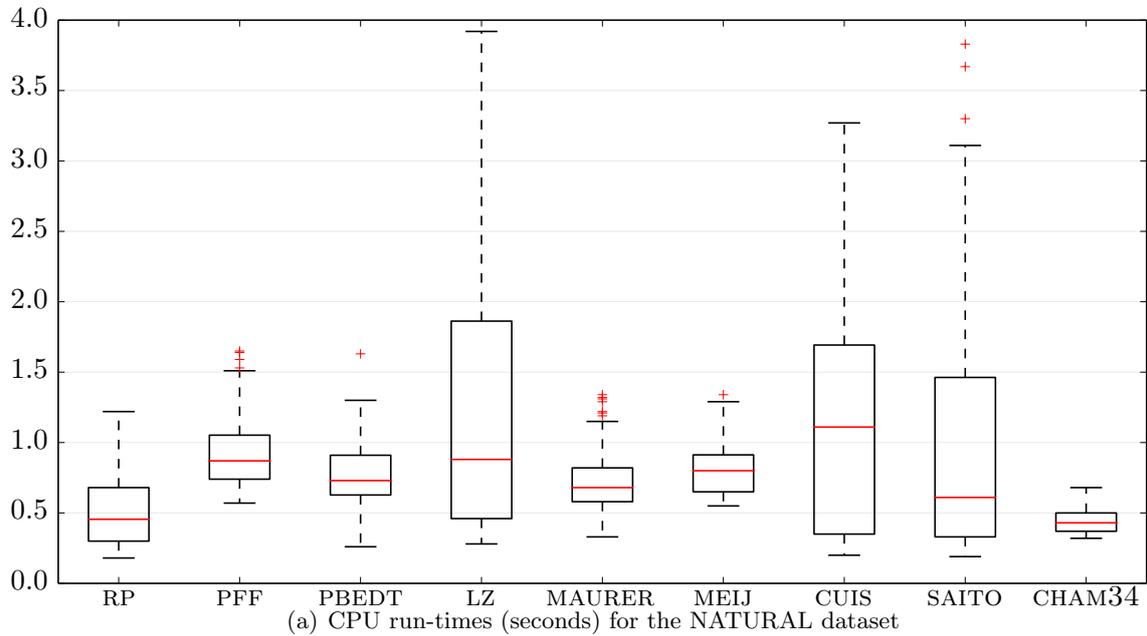


Figure 6-19: Box-plots showing per algorithm median and interquartile range results for the NATURAL dataset

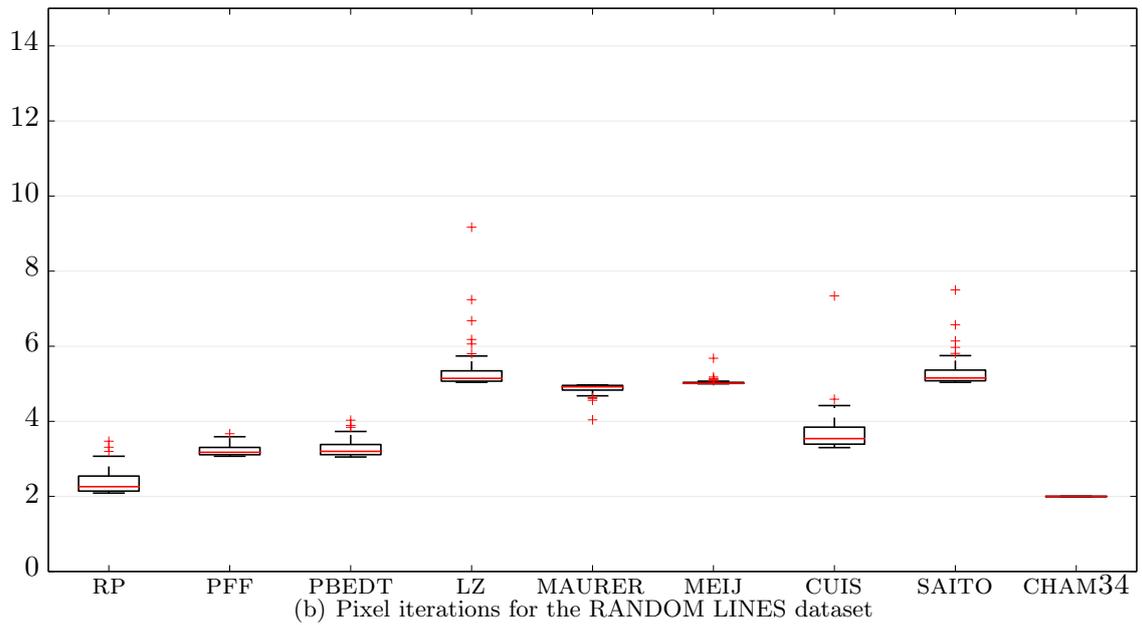
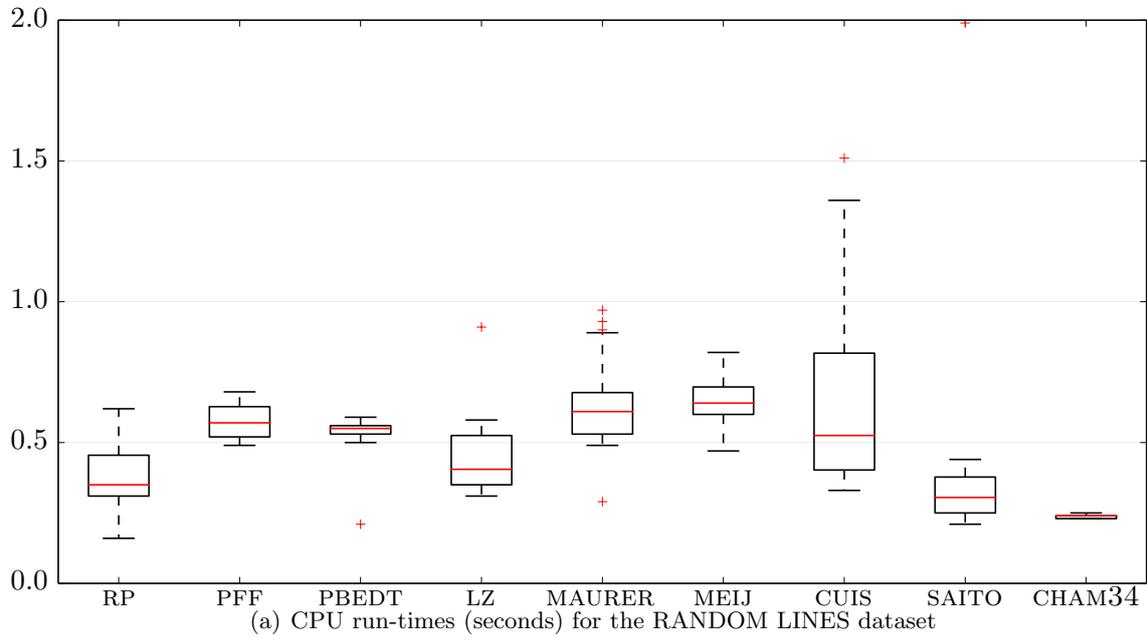


Figure 6-20: Box-plots showing per algorithm median and interquartile range results for the RANDOM LINES dataset

6.5 Conclusion

In this chapter, a new algorithm to calculate the Euclidean distance transform was introduced. The new method is exact and avoids redundant processing of the input/output image meaning that fewer pixel accesses/updates are needed than in previous state-of-the-art methods. The new algorithm is at least as fast as the fastest previous state-of-the-art algorithm and, for many kinds of input, is up to a third faster. For some inputs, the new algorithm is comparable in performance to some methods that compute the inexact Euclidean distance transform.

The algorithm is relatively stable across different image types and retains all of the advantages of previous methods, the most important of which are extensibility to higher dimensional input data (*e.g.* 3-D data such as point clouds) and the ability to take advantage of concurrent processing techniques due to the fact that rows of the input data are parsed independently.

The new algorithm is particularly suited to inputs having a large proportion of foreground points because these values can be ignored; other algorithms use methods that inefficiently recalculate distances to these points. The results of this evaluation show that in all cases where fast calculation of the EDT is sought, the presented algorithm should be preferred over previous methods.

Future research is needed to empirically demonstrate the extensibility of this algorithm, and its attendant performance improvements to higher dimensional input data. The underlying method of scanning each dimension in the input data independently means that the performance improvements demonstrated here on 2-D images should scale exponentially as the number of dimensions in the input data increase. Further research can also investigate how much of a performance improvement can be gained when moving to a non-sequential processing framework.

Chapter 7

Conclusions

Object and pattern recognition is a very broad field. However, most of the computer vision techniques that exist today have been developed with 2-D imagery in mind since this has been the most easily acquired form of data, and our expectations have been high for being able to develop algorithms that can perform at least as well as humans on object recognition tasks using 2-D images such as photographs.

The burgeoning use of mobile mapping technology and the resulting availability of coregistered depth information and 3-D point clouds offers many possibilities to extend these existing techniques, and to develop new techniques with the aim of ultimately improving the efficiency and accuracy of object recognition tasks.

This thesis has explored three popular techniques that are core to the field of object and pattern recognition, and computer vision in general. It has been shown through theoretical and empirical analysis that the new algorithms and modifications to existing algorithms that have been presented in this thesis offer improved object detection / classification accuracy, as well as improvements in efficiency over the existing techniques upon which they build. In addition, this thesis details the significant body of knowledge and mathematical background necessary to develop and evaluate the techniques that have been presented in the main body of this thesis.

In chapter 3, a custom ground-truthing software application was developed and an exercise undertaken to generate the data needed as part of the experimental designs of chapters 4 and 5. This exercise was novel in itself in that it leveraged the talents of individuals with Autism Spectrum Disorder (ASD) to efficiently and accurately label examples of the various object types. Given the limitations encountered in the nature of the datasets used for ground-truthing (see section 7.1), future attempts at ground-truthing new datasets will involve much

more rigorous checking of the data before hand to reduce the possibility of duplicated or poor quality examples. Also presented in chapter 3 was a common mathematical framework to help standardise how the accuracy of different object classification and object detection techniques can be evaluated.

In chapter 4, a new feature extraction method based upon an existing popular feature extraction process – the Histograms of Oriented Gradients (HOG) (Dalal and Triggs, 2005) – was presented. The new method, known as Pro-HOG (Scale Proportionate HOG) takes advantage of explicitly available depth information to encode feature information in fixed dimension scale agnostic descriptors. Pro-HOG was empirically evaluated against the standard HOG algorithm using three different image datasets (two of which having associated coregistered depth information) within a standard object classification framework using both linear and non-linear classifiers generated by a Support Vector Machine (SVM). It was shown that Pro-HOG provides for object classification accuracy that is at least as good as when using HOG. The extraction algorithm developed is more efficient than HOG in time and space, especially in the context of object detection where feature vectors from many thousands of image subregions must be generated. Evaluating the time and space requirements of Pro-HOG versus HOG in the context of object detection is a candidate for future research. Pro-HOG also showed that it is capable of encoding features from the depth maps with floating point accuracy. In future research, performing classification over multiple different object types (as opposed to the binary classification problem) will help to demonstrate the discriminative power of such depth based features. Further comparisons against HOG using an implementation that allows for the sign of the contrast gradient to be encoded should be explored. Repeating the evaluation of HOG by Dalal and Triggs (2005) against the INRIA person dataset using Pro-HOG will also help to further validate its empirical accuracy.

In chapter 5, an existing 2-D object detection and localisation technique – the Class Specific Hough Forest (CSHF) (Gall *et al.*, 2009) – was used to show how detection accuracy can be improved using explicitly available depth information without modifying the underlying algorithmic details of the CSHF. Three different extensions were evaluated against two different datasets having coregistered depth information. It was shown that incorporating scaling into the descriptor extraction process and implicitly scaling the dimensions of the objects being detected helps to improve overall detection accuracy. This was shown using fixed patch dimensions of 0.5×0.5 metres. Future research should investigate the impact of varying these dimensions for different object types on detection accuracy. In addition, it was shown that a very simple (ten dimensional) feature vector generated from the depth information can be competitive in terms of accuracy with much higher dimensional features extracted from the 2-D colour imagery. Future research should investigate whether combining depth and colour based features within a single CSHF improves detection accuracy. In addition,

different parameterisations of the depth based features should be researched since the more or less detailed samplings of the surface morphology of objects may be more or less suited to particular categories of object. Contrary to expectations, the depth weighting extension was empirically found to decrease rather than increase object detection accuracy.

Finally, chapter 6 introduced a new algorithm for generating the Euclidean Distance Transform (EDT) which is a fundamental algorithm used in the matching of Deformable Parts Based Models (DPBM) (Felzenszwalb and Huttenlocher, 2005) as well as many other areas of computer vision and image processing. The method is based upon earlier state-of-the-art algorithms and performance gains are effected by removing certain unnecessary computations to reduce the average number of times the pixels in the input images are parsed. Fast and exact methods of computing the distance transform offer reduced computational overheads when performing object detection using DPBMs. Importantly, the method also theoretically allows for the efficient and exact calculation of distance transforms using input data having greater than two dimensions. This is not demonstrated empirically in this thesis but is a candidate for future research along with an empirical assessment of the performance advantages possible by moving the algorithm to a concurrent processing framework.

7.1 Limitations

Most of the difficulties experienced in producing the experimental results presented in this thesis stemmed from the difficulties in acquiring data of sufficient quality to evaluate the techniques developed in chapters 4 and 5. Paradoxically, even though mobile mapping is seeing much wider use, many of the datasets being generated are by private organisations and as such are not being made freely (or easily) available for research outside of those organisations. It has only been through industry partnerships that it has been possible to use the specific Earthmine and AAM datasets used in this thesis.

The nature of the extensions in chapters 4 and 5 depend upon the fact that existing 2-D only techniques are not intrinsically independent of scale and that the new techniques (Pro-HOG in chapter 4, and the depth extensions to the CSHF in chapter 5) are intrinsically independent of scale because they leverage explicitly available depth information. In order to show that the scale independent nature of these techniques enhances their performance over existing methods, a data environment having good variance in scale is needed.

For use in the evaluations of classification and detection accuracy, it was necessary to undertake a ground-truthing exercise on the Earthmine and AAM provided datasets. As described

in section 3.4, this entailed developing a custom software application and associated process that was undertaken by two volunteers. The aim of the task was to produce object datasets expressing wide intraclass variance in both appearance and scale. Unfortunately, due to the limitations of the mobile mapping process, and the characteristics of the environments being mapped, it became clear that most objects of a particular type in either the Earthmine or the AAM provided datasets presented only within a limited range of scales. Using these ground-truth datasets, this meant that it was difficult in the empirical evaluations of chapters 4 and 5 to show that the new techniques presented in those chapters effected significant improvements in object classification or detection accuracy over the existing methods.

Aside from problems of scale, the datasets were also unable to provide both high quality colour and depth information. Where high quality colour based features could be generated from the Earthmine data, its poor quality depth maps hindered the scale invariant aspects of the new algorithms making the new techniques appear less capable than they may otherwise have appeared with more accurate coregistered depth data available. In the case of the AAM data, colour based feature extractors were less able to meaningfully encode the salient characteristics of objects (or their parts) because of the poor quality coregistration of the colour data to the generated point clouds. This severely limited the range of object classification or detection accuracy achievable by any technique (scale independent or not) thereby also reducing the range of possible improvements in accuracy observable due to any particular technique.

Even with these limitations however, the presented results provide preliminary indications that the algorithms and techniques developed in this thesis offer routes to improved object classification and detection accuracy through the integration of depth data with existing 2-D image based techniques. Moreover, even where accuracy is comparable to existing techniques, significant gains in computing and memory efficiency can be achieved by avoiding the need to scale image subregions, or to conduct detection over multiple image scales. Future research should attempt to verify the utility of the techniques presented in chapters 4 and 5 of this thesis by evaluating them over new datasets having much higher quality colour and depth information, and a much more varied range of object scales.

Bibliography

- 3D Laser Mapping (2015). 3DLM StreetMapper. <http://www.3dlasermapping.com/streetmapper/>. Last Accessed: 2015-09-01.
- Aggarwal, J. and Ryoo, M. (2011). Human activity analysis: A review. *ACM Comput. Surv.*, **43**(3), 16:1–16:43.
- Akmal Butt, M. and Maragos, P. (1998). Optimum design of chamfer distance transforms. *Image Processing, IEEE Transactions on*, **7**(10), 1477–1484.
- Alexe, B., Deselaers, T., and Ferrari, V. (2010). What is an object? In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 73–80.
- Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, **9**(7), 1545–1588.
- Baatz, G., Köser, K., Chen, D., Grzeszczuk, R., and Pollefeys, M. (2010). Handling urban location recognition as a 2d homothetic problem. *Computer Vision/ECCV 2010*, pages 266–279.
- Bailey, D. (2005). An efficient euclidean distance transform. In R. Klette and J. uni, editors, *Combinatorial Image Analysis*, volume 3322 of *Lecture Notes in Computer Science*, pages 394–408. Springer Berlin Heidelberg.
- Bailey, D. G. (2012). Accelerating the distance transform. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand, IVCNZ '12*, pages 162–167, New York, NY, USA. ACM.
- Ballard, D. H. (1981). Generalizing the Hough Transform to Detect Arbitrary Shapes. *Pattern Recognition*, **13**(2), 111–122.
- Bar, M., Kassam, K. S., Ghuman, A. S., Boshyan, J., Schmid, A. M., Dale, A. M., Hmlinen, M. S., Marinkovic, K., Schacter, D. L., Rosen, B. R., and Halgren, E. (2006). Top-down facilitation of visual recognition. *Proceedings of the National Academy of Sciences of the United States of America*, **103**(2), 449–454.

- Barinova, O., Lempitsky, V., and Kholi, P. (2012). On Detection of Multiple Object Instances Using Hough Transforms. *Transactions on Pattern Analysis and Machine Intelligence*, **34**(9), 1773–1784.
- Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., and Wolf, H. C. (1977). Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'77*, pages 659–663, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Barrow, H. G., Tenenbaum, J. M., and Pro, I. (1981). Computational vision. *Proceedings of the IEEE*, **69**(5), 572–595.
- Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, **110**(3), 346 – 359. Similarity Matching in Computer Vision and Multimedia.
- Bo, L., Lai, K., Ren, X., and Fox, D. (2011). Object Recognition with Hierarchical Kernel Descriptors. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Colorado Springs, USA.
- Borck, M., Palmer, R., West, G., and Tan, T. (2014). Using depth maps to find interesting regions. In *Region 10 Symposium, 2014 IEEE*, pages 62–67.
- Borenstein, E. and Ullman, S. (2008). Combined top-down/bottom-up segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **30**(12), 2109–2125.
- Borgefors, G. (1986). Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, **34**(3), 344 – 371.
- Borgefors, G. (1988). Hierarchical chamfer matching: a parametric edge matching algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **10**(6), 849–865.
- Breu, H., Gil, J., Kirkpatrick, D., and Werman, M. (1995). Linear time euclidean distance transform algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **17**(5), 529–533.
- Bülthoff, H. H. and Edelman, S. (1992). Psychophysical support for a two-dimensional view interpolation theory of object recognition. *Proceedings of the National Academy of Sciences*, **89**(1), 60–64.
- Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, **2**(2), 121–167.
- Campbell, R. J. and Flynn, P. J. (2001). A Survey Of Free-Form Object Representation and Recognition Techniques. *Computer Vision and Image Understanding*, **81**(2), 166–210.

- Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **PAMI-8**(6), 679–698.
- Chen, Y., Norton, D., McBain, R., Ongur, D., and Heckers, S. (2009). Visual and cognitive processing of face information in schizophrenia: Detection, discrimination and working memory. *Schizophrenia Research*, **107**(1), 92 – 98.
- Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, IJCAI'11*, pages 1237–1242. AAAI Press.
- Crow, F. C. (1984). Summed-area tables for texture mapping. *SIGGRAPH Comput. Graph.*, **18**(3), 207–212.
- Crowley, J. L., Riff, O., and Piater, J. H. (2002). Fast computation of characteristic scale using a half-octave pyramid. In *Scale-Space Theories in Computer Vision, 2002. International Conference on*.
- Cuisenaire, O. (1999). *Distance transformations: fast algorithms and applications to medical image processing*. Ph.D. thesis, Universit catholique de Louvain (UCL), Louvain-la-Neuve, Belgium.
- Cuisenaire, O. and Macq, B. (1999). Fast euclidean distance transformation by propagation using multiple neighborhoods. *Computer Vision and Image Understanding*, **76**(2), 163 – 172.
- Dalal, N. and Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893. IEEE.
- Danielsson, P.-E. (1980). Euclidean distance mapping. *Computer Graphics and Image Processing*, **14**(3), 227 – 248.
- Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, **40**(2), 5:1–5:60.
- Dickinson, S. J., Metaxas, D., and Pentland, A. (1997). The Role of Model-Based Segmentation in the Recovery of Volumetric Parts From Range Data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **19**(3), 259–267.
- Divvala, S., Hoiem, D., Hays, J., Efros, A., and Hebert, M. (2009). An empirical study of context in object detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1271–1278.

- Dollar, P., Appel, R., Belongie, S., and Perona, P. (2014). Fast feature pyramids for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **36**(8), 1532–1545.
- Eaton, R., Stevens, M., McBride, J., Foil, G., and Snorrason, M. (2006). A systems view of scale space. In *Computer Vision Systems, 2006 ICVS '06. IEEE International Conference on*, pages 3–3.
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, **88**(2), 303–338.
- Everingham, M., Eslami, S., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, **111**(1), 98–136.
- Fabbri, R. (2013). Animal - an imaging library. <http://sourceforge.net/projects/animal/>. Last Accessed: 2013-04-20.
- Fabbri, R., Costa, L. D. F., Torelli, J. C., and Bruno, O. M. (2008). 2d euclidean distance transform algorithms: A comparative survey. *ACM Comput. Surv.*, **40**(1), 2:1–2:44.
- Fan, T.-J., Medioni, G., Nevatia, R., Transactions, I., Pattern, O. N., and Intelligence, M. (1989). Recognizing 3-D Objects Using Surface Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**(11), 1140–1157.
- Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science.
- Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial Structures for Object Recognition. *International Journal of Computer Vision*, **61**(1), 55–79.
- Felzenszwalb, P. F., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.
- Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(9), 1627–1645.
- Fidler, S. and Leonardis, A. (2007). Towards Scalable Representations of Object Categories: Learning a Hierarchy of Parts. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

- Fischler, M. and Elschlager, R. (1973). The representation and matching of pictorial structures. *Computers, IEEE Transactions on*, **C-22**(1), 67 – 92.
- Gall, J., Lempitsky, V., and Eth, Z. (2009). Class-Specific Hough Forests for Object Detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1022–1029. IEEE.
- Gao, T., Packer, B., and Koller, D. (2011). A segmentation-aware object detection model with occlusion handling. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1361–1368, Colorado Springs, USA. IEEE.
- Gehler, P. and Nowozin, S. (2009). On Feature Combination for Multiclass Object Classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 221–228. IEEE.
- Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361.
- Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587.
- Gould, S., Baumstarck, P., Quigley, M., Ng, A. Y. A., and Koller, D. (2008). Integrating Visual and Range Data for Robotic Object Detection. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*, Marseille, France. Andrea Cavallaro and Hamid Aghajan.
- Guan, W. and Ma, S. (1998). A list-processing approach to compute voronoi diagrams and the euclidean distance transform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **20**(7), 757–761.
- Harris, C. and Stephens, M. (1988). A combined corner and edge detector. *Alvey vision conference*, pages 147–152.
- He, N., Cao, J., and Song, L. (2008). Scale Space Histogram of Oriented Gradients for Human Detection. In *Information Science and Engineering, 2008. ISISE '08. International Symposium on*, volume 2, pages 167–170. Ieee.
- Hirata, T. (1996). A unified linear-time algorithm for computing distance maps. *Inf. Process. Lett.*, **58**(3), 129–133.
- Holt, B. and Bowden, R. (2012). Static Pose Estimation from Depth Images using Random Regression Forests and Hough Voting. In *Proceedings of the 7th International Joint*

- Conference on Computer Vision Theory and Applications (VISIGRAPP) (Revised Selected Papers)*, pages 399–410. Springer.
- Hough, P. V. C. (1962). *Methods and Means for Recognizing Complex Patterns*.
- Huang, C.-R., Chen, C.-S., and Chung, P.-C. (2008). Contrast context histograman efficient discriminating local descriptor for object recognition and image matching. *Pattern Recognition*, **41**(10), 3071 – 3077.
- Huber, D., Kapuria, A., Donamukkala, R., and Hebert, M. (2004). Parts-based 3D Object Classification. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004*, volume 2, pages 82–89. IEEE.
- Huttenlocher, D., Klanderman, G., and Rucklidge, W. (1993). Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **15**(9), 850–863.
- Itti, L. and Koch, C. (2001). Computational modelling of visual attention. *Nat Rev Neurosci*, **2**(3), 194–203.
- John, G. S., West, G. A. W., and Lazarescu, M. (2010). Part Based Recognition of Pedestrians Using Multiple Features and Random Forests. In *Proceedings of the International Conference on Digital Image Computing Techniques and Applications (DICTA)*, pages 363–368. IEEE.
- Johnson, A. E. and Hebert, M. (1999). Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **21**(5), 433–449.
- Kim, E. and Medioni, G. (2011). 3d object recognition in range images using visibility context. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3800–3807.
- Kittler, J., Hatef, M., Duin, R. P. W., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**(3), 226–239.
- Knopp, J., Prasad, M., Willems, G., Timofte, R., and Van Gool, L. (2010). Hough Transforms and 3D SURF for Robust Three Dimensional Classification. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, pages 589–602. Springer.
- Krinidis, S. (2012). Fast 2-d distance transformations. *Image Processing, IEEE Transactions on*, **21**(4), 2178–2186.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger,

- editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.
- Kruger, D., Stolkin, R., Blum, A., and Briganti, J. (2007). Optimal auv path planning for extended missions in complex, fast-flowing estuarine environments. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4265–4270.
- Kumar, V. and Patras, I. (2010). A Discriminative Voting Scheme for Object Detection using Hough Forests. In *Proceedings of the BMVC 2010 UK Postgraduate Workshop*, pages 3.1–3.10. BMVA Press.
- Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, **1**(4), 541–551.
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.
- Leibe, B., Leonardis, A., and Schiele, B. (2004). Combined Object Categorization and Segmentation With An Implicit Shape Model. In *Proceedings of the ECCV Workshop on Statistical Learning in Computer Vision*, pages 17–32. Springer.
- Leibe, B., Leonardis, A., and Schiele, B. (2006). An Implicit Shape Model for Combined Object Categorization and Segmentation. *Toward Category-Level Object Recognition*, pages 508–524.
- Leibe, B., Leonardis, A., and Schiele, B. (2007). Robust Object Detection with Interleaved Categorization and Segmentation. *International Journal of Computer Vision*, **77**(1-3), 259–289.
- Li, J. and Allinson, N. M. (2008). A comprehensive review of current local features for computer vision. *Neurocomputing*, **71**(10-12), 1771–1787.
- Li, X. and Guskov, I. (2007). 3D object recognition from range images using pyramid matching. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–6. Ieee.
- Lin, T., Palmer, R., Xia, J., and McMeekin, D. (2014). Automatic generation of station catchment areas: A comparison of euclidean distance transform algorithm and location-allocation methods. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2014 11th International Conference on*, pages 963–967.

- Logothetis, N., Pauls, J., Blthoff, H., and Poggio, T. (1994). View-dependent object recognition by monkeys. *Current Biology*, **4**(5), 401 – 414.
- Lotufo, R. and Zampirolli, F. (2001). Fast multidimensional parallel euclidean distance transform based on mathematical morphology. In *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on*, pages 100–105.
- Lowe, D. (1999). Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2.
- Lowe, D. G. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, **31**(3), 355–395.
- Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, **60**(2), 91–110.
- Maji, S., Berg, A., and Malik, J. (2008). Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.
- Maldonado-Bascon, S., Lafuente-Arroyo, S., Gil-Jimenez, P., Gomez-Moreno, H., and Lopez-Ferreras, F. (2007). Road-sign detection and recognition based on support vector machines. *Intelligent Transportation Systems, IEEE Transactions on*, **8**(2), 264–278.
- Malisiewicz, T., Gupta, A., and Efros, A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 89–96.
- Marr, D. C. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. WH Freeman.
- Marsland, S. (2009). *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st edition.
- Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, **22**(10), 761 – 767. British Machine Vision Computing 2002.
- Maurer, C.R., J., Qi, R., and Raghavan, V. (2003). A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **25**(2), 265–270.

- Mechelli, A., Price, C. J., Friston, K. J., and Ishai, A. (2004). Where bottom-up meets top-down: Neuronal interactions during perception and imagery. *Cerebral Cortex*, **14**(11), 1256–1265.
- Meijster, A., Roerdink, J., and Hesselink, W. (2000). A general algorithm for computing distance transforms in linear time. In J. Goutsias, L. Vincent, and D. Bloomberg, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18 of *Computational Imaging and Vision*, pages 331–340. Springer US.
- Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **27**(10), 1615–1630.
- Mikolajczyk, K., Leibe, B., and Schiele, B. (2006). Multiple Object Class Detection with a Generative Model. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 26–36. IEEE.
- Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, **38**(8), 114–117.
- Moosmann, F., Nowak, E., and Jurie, F. (2008). Randomized Clustering Forests for Image Classification. *Transactions on Pattern Analysis and Machine Intelligence*, **30**(9), 1632–1646.
- Murphy, K. P., Torralba, A., and Eaton, D. (2006). Object detection and localization using local and global features. *Toward Category-Level Object Recognition*, pages 382–400.
- NASA/JPL-Caltech (2014). Photojournal. <http://photojournal.jpl.nasa.gov/targetFamily/mars>. Last Accessed: 2014-01-10.
- Navalpakkam, V. and Itti, L. (2006). An integrated model of top-down and bottom-up attention for optimizing detection speed. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2049–2056.
- Nevatia, R. and Binford, T. O. (1977). Description and Recognition of Curved Objects. *Artificial Intelligence*, **1**(8), 77–98.
- Ng, A. Y. and Jordan, M. I. (2001). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14, Proceedings of the 2001 Conference*, volume 2, pages 841–848.
- Nister, D., Stewenius, H., Nist, D., Stew, H., Nistér, D., and Stewénius, H. (2006). Scalable Recognition with a Vocabulary Tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168.

- Ogniewicz, R. L. and Kübler, O. (1995). Voronoi tessellation of points with integer coordinates: Time-efficient implementation and online edge-list generation. *Pattern Recogn.*, **28**(12), 1839–1844.
- Oliva, A., Torralba, A., Castelano, M., and Henderson, J. (2003). Top-down control of visual attention in object detection. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 1, pages I-253–6 vol.1.
- Opelt, A., Pinz, A., Fussenegger, M., and Auer, P. (2006). Generic Object Recognition with Boosting. *IEEE transactions on pattern analysis and machine intelligence*, **28**(3), 416–431.
- Osuna, E., Freund, R., and Girosi, F. (1997). Training support vector machines: an application to face detection. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 130–136.
- Ott, P. and Everingham, M. (2011). Shared Parts for Deformable Part-Based Models. *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1513–1520.
- Ozuysal, M., Lepetit, V., and Fua, P. (2009). Pose estimation for category specific multiview object localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 778–785. Ieee.
- Palmer, R., Borck, M., West, G., and Tan, T. (2012). Intensity and Range Image Based Features for Object Detection in Mobile Mapping Data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 315–320.
- Palmer, R., West, G. A. W., and Tan, T. (2012). Scale Proportionate Histograms of Oriented Gradients for Object Detection in Co-Registered Visual and Range Data. In *Proceedings of the International Conference on Digital Image Computing Techniques and Applications (DICTA)*. IEEE.
- Palmer, R., West, G., and Tan, T. (2013). Using depth to extend randomised hough forests for object detection and localisation. In *Digital Image Computing: Techniques and Applications (DICTA), 2013 International Conference on*, pages 1–8.
- Palmese, M. and Trucco, A. (2008). From 3-d sonar images to augmented reality models for objects buried on the seafloor. *Instrumentation and Measurement, IEEE Transactions on*, **57**(4), 820–828.
- Pipitone, F. and Adams, W. (1993). Rapid recognition of freeform objects in noisy range images using tripod operators. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 715–716. IEEE Comput. Soc. Press.

- Point Grey Research, Inc (2015). Ladybug3 Spherical Image Capture System. <https://www.ptgrey.com/ladybug3-360-degree-firewire-spherical-camera-systems>. Last Accessed: 2015-09-01.
- Pontil, M. and Verri, A. (1998). Support vector machines for 3d object recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **20**(6), 637–646.
- Porikli, F. (2005). Integral histogram: a fast way to extract histograms in cartesian spaces. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 829–836 vol. 1.
- Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8.
- Rapus, M., Munder, S., Baratoff, G., and Denzler, J. (2008). Pedestrian recognition using combined low-resolution depth and intensity images. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 632–636.
- Rematas, K. and Leibe, B. (2011). Efficient Object Detection and Segmentation with a Cascaded Hough Forest ISM. In *Proceedings of the International Conference on Computer Vision Workshops*, pages 966–973. IEEE.
- Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, **2**(11), 1019–25.
- Rosenfeld, A. and Pfaltz, J. (1968). Distance functions on digital pictures. *Pattern Recognition*, **1**(1), 33 – 61.
- Rosenfeld, A. and Pfaltz, J. L. (1966). Sequential operations in digital picture processing. *J. ACM*, **13**(4), 471–494.
- Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *In European Conference on Computer Vision*, pages 430–443.
- Rowley, H., Baluja, S., and Kanade, T. (1998). Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **20**(1), 23–38.
- Saito, T. and Toriwaki, J.-I. (1994). New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications. *Pattern Recognition*, **27**(11), 1551 – 1565.
- Salti, S., Tombari, F., and Di Stefano, L. (2010). On the use of Implicit Shape Models for Recognition of Object Categories in 3D Data. In *Proceedings of the 10th Asian Conference on Computer Vision (ACCV)*, pages 653–666. Springer.

- Schroff, F., Criminisi, A., and Zisserman, A. (2008). Object Class Segmentation using Random Forests. In *Proceedings of the British Machine Vision Conference*, pages 1—1.
- Schlkopf, B., Burges, C., and Vapnik, V. (1995). Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery & Data Mining, Menlo Park*, pages 252–257. AAAI Press.
- Sengupta, K. and Boyer, K. L. (1995). Organizing Large Structural Modelbases. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **17**(4), 321–332.
- Shih, F.-C. and Mitchell, O. R. (1992). A mathematical morphology approach to euclidean distance transformation. *Image Processing, IEEE Transactions on*, **1**(2), 197–204.
- Shih, F. Y. and Wu, Y.-T. (2004). Fast euclidean distance transformation in two scans using a 3x3 neighborhood. *Computer Vision and Image Understanding*, **93**(2), 195 – 205.
- Shotton, J., Blake, A., and Cipolla, R. (2005). Contour-based learning for object detection. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 503–510 Vol. 1.
- Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., and Freeman, W. T. (2005). Discovering objects and their location in images. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 370–377. Ieee.
- Snavely, N., Simon, I., Goesele, M., Szeliski, R., and Seitz, S. (2010). Scene reconstruction and visualization from community photo collections. *Proceedings of the IEEE*, **98**(8), 1370–1390.
- Steder, B., Rusu, R. B., Konolige, K., and Burgard, W. (2010). NARF: 3D Range Image Features for Object Recognition. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS): Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics*. IEEE.
- Stein, F. and Medioni, G. (1992). Structural indexing: Efficient 3D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**(2), 125–145.
- Sun, M., Bradski, G., Xu, B.-X., and Savarese, S. (2010). Depth-Encoded Hough Voting for Joint Object Detection and Shape Recovery. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, volume 5 of *ECCV'10*, pages 658–671. Springer.
- Torralba, A. and Sinha, P. (2001). Statistical context priming for object detection. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 763–770 vol.1.

- Tuytelaars, T. and Mikolajczyk, K. (2007). Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision*, **3**(3), 177–280.
- van de Sande, K., Gevers, T., and Snoek, C. (2010). Evaluating color descriptors for object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **32**(9), 1582–1596.
- Vedaldi, A., Gulshan, V., Varma, M., and Zisserman, A. (2009). Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613.
- Wang, J. and Tan, Y. (2013). Efficient euclidean distance transform algorithm of binary images in arbitrary dimensions. *Pattern Recognition*, **46**(1), 230 – 242.
- Wang, T., He, X., and Barnes, N. (2012). Learning hough forest with depth-encoded context for object detection. In *Digital Image Computing Techniques and Applications (DICTA), 2012 International Conference on*, pages 1–8.
- Wei, X., Phung, S. L. S., and Bouzerdoum, A. (2011). Pedestrian sensing using time-of-flight range camera. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 43–48.
- Witkin, A. P. (1984). Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84.*, volume 9, pages 150–153.
- Wu, C., Clipp, B., Li, X., Frahm, J.-M., and Pollefeys, M. (2008). 3D model matching with Viewpoint-Invariant Patches (VIP). In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 0, pages 1–8, Los Alamitos, CA, USA. IEEE Computer Society.
- Yamazaki, M. and Fels, S. (2009). Local Image Descriptors Using Supervised Kernel ICA. *IEICE Transactions on Information and Systems*, **E92-D**(9), 1745–1751.
- Zhang, Y. and Chen, T. (2010). Weakly Supervised Object Recognition and Localization with Invariant High Order Features. In *Proceedings of the British Machine Vision Conference 2010*, pages 47.1–47.11. British Machine Vision Association.
- Zhang, Z., Warrell, J., and Torr, P. H. S. (2011). Proposal Generation for Object Detection using Cascaded Ranking SVMs. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1497–1504, Colorado Springs, USA.
- Zhao, L. and Thorpe, C. (2000). Stereo and neural network-based pedestrian detection. *Intelligent Transportation Systems, IEEE Transactions on*, **1**(3), 148–154.

Zhu, L., Chen, Y., Yuille, A., and Freeman, W. (2010). Latent hierarchical structural learning for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1062–1069.

Zhu, S.-C. and Mumford, D. (2006). A stochastic grammar of images. *Found. Trends. Comput. Graph. Vis.*, **2**(4), 259–362.

Every reasonable effort has been made to acknowledge the owners of copyright material. I would be pleased to hear from any copyright owner who has been omitted or incorrectly acknowledged.

Appendix A

Glossary of Terms

Appearance Based Model: A *Model* that is concerned primarily with encoding how a type is projected into the image domain.

Bounding Box or Rectangle: The delineation of an object or feature of interest is often specified as a rectangular sub-region of an image. This defines an image extract. Bounding boxes are used because they are simple to define, although when defining an object's bounds, they often also include background image information that is not specific to the object, but may be helpful contextually for detection tasks because certain kinds of objects are often present against particular backgrounds. For example, cars often appear with road surfaces as part of their background.

Classification: Estimating the value of a discrete random variable Y as the result of evaluating a discriminating function that takes X as input. X may be either continuous or discrete. See *Regression*.

Clustering: A form of *Unsupervised Learning* where data are grouped within some *Feature Space* according to their similarity in that space.

Codebook: Within a *Model*, specific high level characteristics about an object type may be particularly prevalent. The canonical encoding of this characteristic within a *Model* comprises a codebook entry. Codebook entries may be considered as models within models since they encode expected variation for a simpler component.

Cross Validation: A process for dividing a ground-truthed dataset into training and validation subsets, using the training set to build a classification or regression function, and evaluating the correctness of that function on the validation set. Typically, the process iterates using different training and validation subsets until every example in the dataset has been used for both training and validation. In N -fold cross validation, the dataset is divided into N subsets and each of the N subsets is used as the validation set with the remaining $N - 1$ subsets being used for training. The value of N can range from two up to the size of the dataset itself (in

which case the scheme is termed *Leave-One-Out Cross Validation*.

Decision Tree: An algorithm that encodes a branching sequence of questions to be asked of some input data (usually a *Feature Vector*). Input data are provided to the tree at the *root* node and the resulting evaluation of each node's query about its input data determines the child node that the input data are sent to next. The last node in the sequence is termed a *leaf* node. Each node, except the leaf nodes can have multiple children. Decision Trees are directed acyclic graphs. A common form of Decision Tree encodes a binary predicate decision in each node (except the leaf nodes) resulting in a binary tree having a depth of $\log_2 N$ for N nodes. Decision trees are typically used as one way of encoding classifier functions.

Depth Map / Image: An image where each pixel is a value that represents the distance from the camera plane to the corresponding point in a scene.

Dictionary: See *Codebook*.

Discriminative Modelling: A discriminative model is primarily concerned with accurately estimating and encoding the boundaries between types or categories.

Dynamic Analysis: An analysis of the computational complexity of an algorithm as measured against specific input data. This is distinct from a *Static Analysis* which provides a theoretical analysis of the run-time computational complexity of an algorithm in the worst, best or average case by counting the number of source code statements executed in these scenarios.

Feature: Either a particular high level object of interest (*e.g.* a street sign) or a characteristic of some object in some scene data (*e.g.* a building edge). A feature encodes a specific characteristic of an image, image subregion, or object measured along a particular subset of dimensions (see *Local versus Global*). Scalar encodings are produced by measurements along a single dimension. Vector encodings are produced by measurements over multiple dimensions. These scalars or vectors can be concatenated into feature vectors or feature descriptors (see section 2.1).

Feature Space: When being used to connote specific characteristics of an image, feature values are often represented by vectors. After extracting all features from a location in some data (*e.g.* an image), the value of those features can be represented as a point in N -dimensional space where N is the length of the feature vector giving the concatenation of all the scalar values representing the features of interest from that location in the data. See section 2.1.

Geometry Based Model: A *Model* that is concerned primarily with encoding how a type is structured. Such models are often used to match objects to templates through translations, rotations, scalings, and deformations of parts.

Generative Modelling: A generative model encodes sufficient information about a type that examples of the type can be generated (if only notionally) from the model.

Ground-Truth: The manually validated locations and labels of objects or patterns in data. The process of generating the ground-truth is termed *ground-truthing*. Data that have undergone ground-truthing are said to be *ground-truthed* or labelled.

Histograms of Oriented Gradients (HOG): A specific kind of feature that encodes contrast information about an object or image subregion. See section 4.1.1.

Integral Image: A matrix whereby the sum over any arbitrary rectangular subregion of the matrix can be evaluated by a simple arithmetic calculation involving the values in the matrix at the four corners of the subregion. See figure 4-1 (page 72).

Inter-class Variability: The degree of variation expressed between the model parameter distributions of different object classes.

Intra-class Variability: The degree of variation expressed between the individual instances (members) of a modelled object class.

Interest Point: Specific kinds of features (*e.g.* edges, corners, particular shapes) that are typically indicative of more interesting parts of a scene or object. Such features may help to encode within a model the distinctiveness about a particular object type, or to indicate locations within a scene that the recognition scheme should prioritise. Also known as *Keypoints*.

Keypoint: See *Interest Point*.

Local versus Global: When used in the context of image features, local connotes a feature of interest at some specific location within the image. Global connotes a feature that is characteristic of the whole image.

Model: An encoding of the essential characteristics of some object type. The accuracy and fidelity of the model depends upon its intended use. There are two varieties of modelling: *Generative* and *Discriminative*. Models encode the degree of allowed variation in the characteristics that parameterise the object type. See *Intra/Inter-class Variability*.

Object Class / Type / Category: Objects can be grouped in accordance with different measures of similarity including function, geometry, or appearance in scenes. A grouping of objects according to these criteria specifies an object class or type. How an object class is encoded is specified by a *Model* of the object type.

Object Classification: Given some predefined list of labels, the association of some example to one of those labels.

Object Detection: The determination that some instance or example of a type is present.

Object(s) of Interest: Specific examples of the object type that the user wishes to find.

Object Localisation: The determination of the location of some object in a scene. Usually undertaken in concert with *Object Detection*.

Object Recognition: The evaluation of a parsed pattern of data as an example of some previously known type (in which case recognition utilises *Object Classification*), or as an example of some previously unknown type.

Parts Based Model: An encoding of a type via a modelling of its component parts.

Precision: Given a method of estimating whether an item is an example of some type, precision is the ratio of correct estimates to the total set of all estimates made. In other words, the proportion of correct estimates.

Recall: Given a method of estimating whether an item is an example of some type, recall is the ratio of correct estimates to the total set of actual examples of the type. In other words, the proportion of discovered examples.

Regression: Estimating the value of a continuous random variable Y as the result of evaluating a function that models the distribution of Y against an input variable X . X may be either continuous or discrete. See *Classification*.

Segmentation: Segmenting an object is the act of either manually or automatically determining the strict bounds of an object.

Salient Region: Those characteristics or elements of an object, or scene that provide more “useful” information to the object recognition task. For example, a homogenous image region is typically less *salient* than an equivalently sized image region have some contrast information. This is because the contrast information encodes extra information the homogenous image region cannot such as orientation (or even scale). Some *Interest Point* detectors are designed to identify specific kinds of such regions.

Scene: The image that contains objects of interest as well as other scene elements that may or may not provide assistance in determining the presence and locations of the objects of interest.

Static Analysis: See *Dynamic Analysis*.

Supervised Learning: An algorithm that takes labelled training data as input and produces a classification / regression function that aims to estimate the class membership of new (unlabelled) instances.

Support Vector Machine (SVM): A type of *Supervised Learning* characterised by its identification of the minimal subset of training data (the Support Vectors) that are sufficient to model a discriminative classification function.

Thresholding: Discretisation of an input into binary values α and β based on a threshold value τ . For some input \mathbf{x} , the thresholding function t is defined as

$$t(\mathbf{x}, \tau) = \begin{cases} \alpha & \text{for } x_i < \tau \\ \beta & \text{for } x_i \geq \tau \end{cases}, \forall x_i \in \mathbf{x}. \quad (\text{A.1})$$

Type I Error: A false positive. Incorrectly identifying an item as an example of some type.

Type II Error: A false negative. Failing to recognise an item as an example of some type.

Unsupervised Learning: An algorithm that takes unlabelled training data as input and produces a classification / regression function that aims to estimate the class membership of new (unlabelled) instances.