School of Civil and Mechanical Engineering

Discrete Path Planning for Convex Polyhedra through Edge-Rolling on a Plane

Ngoc Tam Lam

0000-0001-6478-5802

This thesis is presented for the Degree of

Doctor of Philosophy

of

Curtin University

September 2022

Abstract

The study of convex polyhedra has attracted many researchers in mathematics, sciences, and arts. Most of the previous studies focused on geometrical properties, such as polyhedron unfolding, polyhedral intersection, and pattern combinations. An early discussion of regular polyhedra concerned the five Platonic solids—tetrahedron, cube, octahedron, icosahedron and dodecahedron—traces back to Plato in Timaeus when they were identified as fire, air, earth, water, and quintessence or aether elements. Path planning for polyhedra through edge-rolling has been suggested by several researchers, but not validated, except for solving the rolling-cube puzzle.

The aim of this study was to develop discrete path-planning algorithms for rolling a convex polyhedron from an initial pose (position and orientation) to a goal pose on a plane. This thesis first applied a breath-first search (BFS) based algorithm to edge-roll a tetrahedron, cube, octahedron, icosahedron, and dodecahedron, which consists of 4 regular triangles, 6 regular squares, 8 regular triangles, 20 regular triangles, and 12 regular pentagons, respectively, on prescribed grids. It then applied a randomly explored tree (RRT) based algorithm to edge-roll a truncated icosahedron (the geometry is associated with a soccer ball), which consists of 12 regular pentagons and 20 regular hexagons.

In the first part, polygon tiling was used to generate different grids on the plane for the Platonic solids: square grids for the cube, triangular grids for the tetrahedron,

octahedron and icosahedron, and pentagon grids for the dodecahedron, where the Penrose tiling was chosen from several tiling patterns because it has five-fold symmetry. Two scenarios were considered depending on the size of the search space. If the search space is limited leading to the acceptability of the computation time of collision check and graph expansion of the BFS algorithm, a one-step approach would be used to find the shortest path for each solid to reach both the goal position and goal orientation. On the other hand, if the search space is largely resulting in the failure to find potential paths in a reasonable computational time due to the curse of dimensionality, a two-step approach would be applied: the first step entailed path planning to reach the goal position; the second step generated a closed path so that the goal orientation was achieved.

In the second part, the RRT-based algorithm was applied to the truncated icosahedron on a non-prescribed plane, where both its initial and goal poses were randomly chosen. Sampling poses were generated from the unfolded truncated icosahedron on the plane. The potential poses were chosen outside of the collision zones defined by an offset distance equal to the circumradius of the truncated icosahedron. The proposed algorithm achieved an increased convergence rate by imposing a strong bias to guide the growth of the randomized tree in the direction of the goal position while some other RRT variants searched the whole space, resulting in slow convergence. However, the truncated icosahedron cannot always reach the exact goal position because of the discretised steps. Thus, a path would be only deemed successful if the distance between the truncated icosahedron's centre and the goal centre is less than

the radius of the truncated icosahedron, and the goal orientation is reached.

The BFS-based algorithm traversed the nodes in a graph to construct the shortest paths. Compared with the A* algorithm and its variants, the proposed BFS algorithm performed a complete search by pre-orderly extending the parent node and all neighbours of the child nodes until achieving the goal. It is guaranteed to generate the shortest path at a higher computational cost than the A* variants which sometimes fail to find a feasible path. In this thesis, the search space is limited for path planning for the Platonic solids on prescribed grids, thus the BFS-based algorithm was preferred, and it produced the desired results—shortest paths.

Path planning for the truncated icosahedron on a non-prescribed plane posed a great challenge: the search space is large, which rendered the BFS-based algorithm inadequate. The proposed RRT-based algorithm first constructed a convex hull that included the initial and goal positions of the truncated icosahedron and all the obstacles and then sampled nodes inside this convex hull. In this way, the algorithm can enforce a strong bias guide to expand the randomized tree to the goal position, which increased the convergence rate. In comparison, some RRT variants, such as CL-RRT and RT-RRT*, can generate sub-optimal paths by exploring a large number of nodes, which was computationally expensive. Other RRT variants, such as A*-RRT*, combine RRT* with A* to accelerate the convergence rate. However, the initial path generated by A* took a large number of nodes with a Euclidean distance as the heuristic function prior to guiding the RRT* sampling process.

In summary, this thesis solved the path-planning problem of the Platonic solids and truncated icosahedron through edge-rolling on a plane with obstacle avoidance, which hitherto had not been solved. The BFS-based algorithm found the shortest paths for the Platonic solids on a prescribed plane while the RRT-based algorithm generated feasible paths with efficient tree exploration on a non-prescribed plane. The results can be readily applied to a variety of applications: path planning for general convex polyhedral, dexterous robotic in-hand manipulation, video games, and locomotion of polyhedral tensegrity robots.

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature:

Date: 31/05/2022

Acknowledgments

This thesis is the culmination of many years of pleasant association with Curtin University and the School of Civil and Mechanical Engineering. Financial support for this work was provided by the Curtin International Postgraduate Research Scholarship (CIPRS) and Research Stipend Scholarship.

There are many people who helped me and supported me throughout my time as a Ph.D. student at Curtin University. I want to express my gratitude to my thesis supervisor, Dr. Lei Cui. Dr. Lei is an excellent mentor and significantly influenced my intellectual path. I also want to thank Dr. Lei for providing extremely advice for my career path. I am also immensely grateful to my co-supervisor, Prof. Ian Howard, for his constant support and valuable advice on my research.

I want to thank my colleagues, Thibault Rouillard, Zefang Shen and Pratik Pandya, who always supported and impressed me with their know-how and willingness to discuss the research problems. I am honoured to be a part of this research group.

I want to thank all my friends outside the lab who have been there throughout my time at Curtin University. Special thanks to Thong, San, Tuan, Tung, Nhi, Tin and Duong for all the fun times we have had throughout the years. Graduate school would have been the same. I would like to thank Cheryl for all the support during the Ph.D. program.

Finally, I would like to thank all my family for all the love, support, encouragement and prayers they have sent my way along this journey. To my parents and parents-in-law, thank you so much for always standing by me in all my endeavours and supporting me through every major decision I have made in my life. I hope that I have made you proud. To my sister, thanks for the love, support, and encouragement. To my beloved wife, Diem Phuong, who supported me through all the ups and downs of this journey. I would like to thank you so much. Without you, I would not be where I am today. And to my lovely girls, Minh Thu - Tina and Minh An - Annie, you are my beautiful angels and my inspiration to achieve greatness. So thank you all - for always being there.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Motivation

The path planning problem is to determine a path of an object from its source (initial) position to a destination (goal) position through a workspace populated with obstacles. The obstacles may be either stationary or moving. The desired path is an optimal one with no collisions between the object and the obstacles. The term path planning was developed in many fields, such as robotics, artificial intelligence and control theory. That is why scientists use their definition of this term [1]. There are many different aspects of the path planning problem for a moving object. These include optimal path planning among rectangular obstacles, optimal path finding among weighted regions, and path planning to traverse narrow passages (the 'Piano Movers Problem'). The task is defined with a precise model of a house and a piano as input to an algorithm. The algorithm must determine how to move the piano from one room to another without hitting anything. A robot's path planning is defined in a similar way. However, path planning usually ignores dynamics and other constraints and focuses primarily on the translations and rotations of the controlled object. Recent research in this area also considers other aspects such as uncertainties [2], differential constraints [3] and optimality [4, 5].

The problem of dexterous manipulation of objects, for example, arbitrary relocation and reorientation of rigid bodies by the action of some mechanism, is considered a path planning problem. In some cases, the problem is that of reorienting a large number of parts in random positions and orientations, to a given posture within assembly tolerances. Dexterous manipulation of the position and/or orientation of polyhedral is a nonholonomic behaviour achieved by pushing, tilting, re-grasping, rolling and translating. Rolling polyhedral has been suggested, but not validated in any path planning algorithm, except for solving the rolling-cube puzzles for a cubic dice [6].

Finding a collision-free path from a given starting pose to a goal pose for a polyhedron through edge-rolling around static obstacles is called the discrete path planning problem. Path-finding algorithms are developed from graph search techniques in general. For each node of the graph, there are only a discrete number of choices to decide in which direction to move next. Discrete search algorithms do not actually require an explicit graph to existing. It is enough to have a state transition function f that returns the next state for every possible action to be taken from the current state. Let u denote action and x the current state. The next state obtained after applying action u will be $x' = f(x, u)$. This notation is more generic since it does not require the entire state graph of the problem to be explicitly available, it only requires the moves between states to be known. Function f can represent the possible moves of an agent and as well of any system that is described by states. For example, when solving a Rubik's cube puzzle, the transition function f is well defined because all

the discrete number of possible rotation moves to change the cube from the current state to any adjacent state are known. This thesis focuses on the path-finding algorithm for the convex polyhedron through edge-rolling and surface contact. The discrete path has consisted of curves that indicate the rotation of the centre of the polyhedron from a current state to the next state.

1.2 Aims and Objectives

The aim of this study is to develop discrete path planning algorithms for rolling convex polyhedra to achieve a goal pose (position and orientation) from an initial pose on a plane.

The following objectives have been proposed to fulfill the aims of the thesis.

(a) Analysing the geometrical structure of the Platonic solids and the truncated icosahedron which is representative of a soccer ball structure.

(b) Applying the breadth-first search (BFS) based algorithm to find a path for the Platonic solids through edge-rolling with obstacle avoidance on prescribed grids.

(c) Implementing the path planning for the Platonic solids in two case scenarios. The first scenario is to find a path through edge-rolling from an initial pose

to a goal pose, which is applied for the limited search space. On the other hand, if the search space is large, the shortest path from the initial position to the goal position is achieved without orientation consideration. The BFS-based algorithm then is applied to find a closed path so that the goal orientation is reached.

(d) Applying the rapidly-exploring random tree (RRT) based algorithm to the truncated icosahedron on a non-prescribed plane. By imposing a strong bias to guide the growth of the randomized tree in the direction of the goal position, the convergence rate of the algorithm is increased. Achieving the goal orientation with the closed path would be only supposed successful if the distance between the truncated icosahedron's centre and the goal centre is less than the radius of the truncated icosahedron.

1.3 Thesis Outline

The thesis is organised into 7 chapters; the details are as follows:

Chapter 1 is an introduction to the thesis. It begins with the motivation and background of the path planning algorithms for rolling polyhedra. The aims and objectives of the thesis are then proposed to solve the problem. The outline of the structure of the thesis ends the chapter.

Chapter 2 is an introduction to the mathematical tools used in the thesis. It first describes the rigid body transformations in the $3D$ space before analyzing the rotation matrix which is applied in the path planning algorithms for polyhedra through edge-rolling.

Chapter 3 shows how the study in the thesis fits within the literature by introducing a detailed review of related works and highlighting the gaps in the previous studies. It first provides a comprehensive review of rolling contact and explains how this work proposed the algorithm to tackle the problem. The $3D$ convex polyhedra will be utilised and the two convex polyhedra groups: Platonic solids and the Truncated icosahedron are analysed in the geometrical properties. Finally, it presents the review of path planning methods and highlights two main algorithms used in the thesis: breadth-first search (BFS) and rapidly-exploring random tree (RRT) based algorithms. The chapter provides the summary and gaps in the literature at the end.

Chapter 4 introduces details of the BFS and RRT-based algorithm which will be applied to find the feasible paths for rolling polyhedra. The chapter first reviews the literature on the BFS algorithms and previous applications. Although the BFS algorithm has the disadvantage of computation time, the algorithm can find the optimal rolling path for the convex polyhedra if it exists through the analysis in the chapter (there is no guarantee to find the paths for the general convex polyhedra due to the geometrical complexity). The implementations of the BFS algorithm for the Platonic solids are shown in Chapter 5. In the second part, the chapter presents the

RRT-based algorithm to solve the path planning for the general convex polyhedra through edge-rolling. An overview of the RRT algorithm and its previous academic and industrial applications are presented. A preliminary of the algorithm consistently with the defined environment for a polyhedron edge-rolling is then introduced. The proposed RRT-based algorithm for a convex polyhedron to find the rolling path through edge-contact is presented in detail. Its implementations for the truncated icosahedron are detailed in Chapter 6.

Chapter 5 adopts the BFS-based algorithm from the first part of Chapter 4 to solve the path planning problem for the Platonic solids through edge-rolling on the prescribed grids. First, it introduces the related works of rolling polyhedra and the previous gaps needing to be solved. The detailed properties of Platonic solids are presented belong the discretized grids problems which support the environment for the rolling motions. Finally, the chapter shows the significant results by using the BFS-based algorithm for the Platonic solids to find the shortest paths from an initial pose to a goal pose while avoiding obstacles.

Chapter 6 adopts the RRT-based algorithm from the second part of Chapter 4 to find a closed path for the truncated icosahedron through edge-rolling on a plane while avoiding obstacles. The chapter firstly introduces the related works of the RRT path planning algorithms and their applications. The truncated icosahedron properties are then provided with the rolling contact. The chapter then shows the results of closed path-findings for the truncated icosahedron to reach the goal pose from the start pose with and without obstacles on a plane.

Chapter 7 presents the general conclusion of the thesis. The contributions of the work and the main achievements of the thesis for solving the path planning problems of the convex polyhedra on a plane are highlighted. Future work is also presented.

Chapter 2: Preliminaries

Polyhedra rolling contact through edge-rolling is considered a nonholonomic system in terms of changing the orientation of the objects while maintaining the edge contact. This chapter firstly summarises the concept of the rigid body transform in Euclidean space, then summarises the preliminaries of the transformation matrix used throughout this work.

2.1 Rigid Body Transformations

Transformation is simply defined by changing the position and orientation of a frame attached to an object with respect to a frame attached to another object. A rigid body is a set of points or articles in Euclidean space in which mutual distances are fixed regardless of trajectories at each instant of time, relative to an inertial Cartesian coordinate frame. By the definition, a rigid body transform is a mapping from this set to another subset of the Euclidean space [7]. In this thesis, the triple $(x, y, z) \in \mathbb{R}^3$ represents the set of three orthonormal axes, where each coordinate gives the projection of the object's position onto the corresponding axis. The rigid body can move on a trajectory represented by a parameterized curve $p(t) = (x(t), y(t), z(t)) \in \mathbb{R}^3$.

In the three-dimensional ($3D$) space, the rigid body transforms are composed of translations along $3D$ vectors and rotations around the coordinate axes. A $3D$ transform $\boldsymbol{T}$ is represented by a 4 matrix when the points are defined with homogeneous coordinates.

$$\boldsymbol{T} = \begin{bmatrix} i_1 & j_1 & k_1 & t_x \\ i_2 & j_2 & k_2 & t_y \\ i_3 & j_3 & k_3 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

where $\boldsymbol{i} = [i_1\ i_2\ i_3]^T$, $\boldsymbol{j} = [j_1\ j_2\ j_3]^T$ and $\boldsymbol{k} = [k_1\ k_2\ k_3]^T$ are the unit vectors of the transformed coordinate system, while $\boldsymbol{t} = [t_x\ t_y\ t_z]^T$ is the position of its origin with respect to the original coordinate system.

A rigid motion of an object which is described as a subset $\boldsymbol{O}$ of $\mathbb{R}^3$ is represented by continuous mappings. The generation of Eq. 2.1 is used for mapping the individual points which is represented as

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} i_1 & j_1 & k_1 & t_x \\ i_2 & j_2 & k_2 & t_y \\ i_3 & j_3 & k_3 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.2}$$

2.2 Rotation Matrix

2.2.1 Properties of Rotation Matrices

The orientation of the body can be described through the relative orientation between a fixed coordinate frame ($\boldsymbol{A}$) and the coordinate frame (attached to the body) ($\boldsymbol{B}$). A basic rotation 3×3 matrix $\boldsymbol{R}_{ab} = [\boldsymbol{x}_{ab}\ \boldsymbol{y}_{ab}\ \boldsymbol{z}_{ab}]$, and $\boldsymbol{x}_{ab}, \boldsymbol{y}_{ab}, \boldsymbol{z}_{ab} \in \mathbb{R}^3$. The set of all 3×3 matrices is denoted $\boldsymbol{SO}(3)$ which is a group using the identity matrix $\boldsymbol{I}$. $\boldsymbol{SO}(3)$ is a short form for the special orthogonal group in the three-dimensional space. It is also called the rotation group which represents the rotations around a line or an axis. Every rotation can be represented uniquely by an orthogonal matrix with a unit determinant. A (positive, orthonormal) frame is a list of three mutually perpendicular unit vectors in $\mathbb{R}^3$ which, when viewed as the columns of a 3×3 matrix, lie in the group $\boldsymbol{SO}(3) = \{\boldsymbol{R} : \boldsymbol{R}\boldsymbol{R}^{\boldsymbol{T}} = \boldsymbol{I}_3 \ and \ det(\boldsymbol{R}) = 1\}$.

$\boldsymbol{SO}(3)$ denotes the Special Orthogonal Group and is a subgroup of the Orthogonal Group and the general linear group. To sum up, the matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ is said to represent the attitude of a rigid-body if and only if $R \in \boldsymbol{SO}(3)$ which, in turn, is true when the following two above conditions are satisfied. $\boldsymbol{R}$ is called rotation by inversion if $det(\boldsymbol{R}) = -1$, which does not belong to $\boldsymbol{SO}(3)$ and therefore, will not be considered in this section. The focus of this analysis is $\boldsymbol{R} \in \boldsymbol{SO}(3)$, which satisfies both of the above-mentioned conditions. The main advantage of the $\boldsymbol{R} \in \boldsymbol{SO}(3)$ representation

is that the attitude $\boldsymbol{R}$ is global and unique, implying that each physical orientation of a rigid-body corresponds to a unique rotational matrix.

Rotation matrix has the property of the rotation of the cross product of two vectors is the cross product of the rotation of each of the vectors by $\boldsymbol{R}$ with $\boldsymbol{R}(a \times b) = \boldsymbol{R}(a) \times \boldsymbol{R}(b)$. The basic cross product between two vectors $\vec{a}, \vec{b} \in \mathbb{R}^3$ is denoted as:

$$\vec{a} \times \vec{b} = \begin{vmatrix} i & j & k \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix} = (a_2b_3 - a_3b_2)i + (a_3b_1 - a_1b_3)j + (a_1b_2 - a_2b_1)k \tag{2.3}$$

The rotation matrix $\boldsymbol{R} \in \boldsymbol{SO}(3)$ also presents the transformation which transfers the coordinate of a point from one frame to another frame. Let consider a point q with $q_b = (x_b, y_b, z_b)$ is its coordinate and $x_b, y_b, z_b \in \mathbb{R}^3$ are the projections of q onto the coordinate axes of the body. Then, the coordinate of q is defined as $q_a = \boldsymbol{x}_{ab}x_b + \boldsymbol{y}_{ab}y_b + \boldsymbol{z}_{ab}z_b = \boldsymbol{R}_{ab}q_b$.

2.2.2 Rodrigues Rotation Matrix

In this thesis, the Rodrigues' rotation matrix from the axis-angle representation [8] was used to calculate the orientation of the convex polyhedra (the Platonic solids and truncated icosahedron are considered in this thesis) from a current position to

the next position after edge-rolling. A unit vector $\boldsymbol{\omega} \in \mathbb{R}^3$ and a rotation angle β, which are specific with respect to each polyhedron, were used to generate a rotation matrix $\boldsymbol{R} \in \boldsymbol{SO}(3)$ (Eq (2.5)). In a three-dimensional Euclidean coordinate system, an axis-angle representation is given by a unit vector $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ and an angle β. The rotation angle β of each polyhedron is the supplementary angle of the dihedral angle α, which is the angle between two intersecting faces (more detail in Fig. 3.7 - Chapter 3).

Let S be the Rodrigues' vector of this rotation with the angle θ about an orientated axis through the origin of the coordinates (Fig. 2.1). The vector P_1P is tangent to the arc described by point P_1 as it rotates about the axis. In this figure, the diagonal P_1P_2 and OP that are perpendicular to each other, its intersection at the point P_m. A half rotation angle can be obtained by $\tan\frac{\theta}{2} = \frac{P_1P_m}{OP_m}$

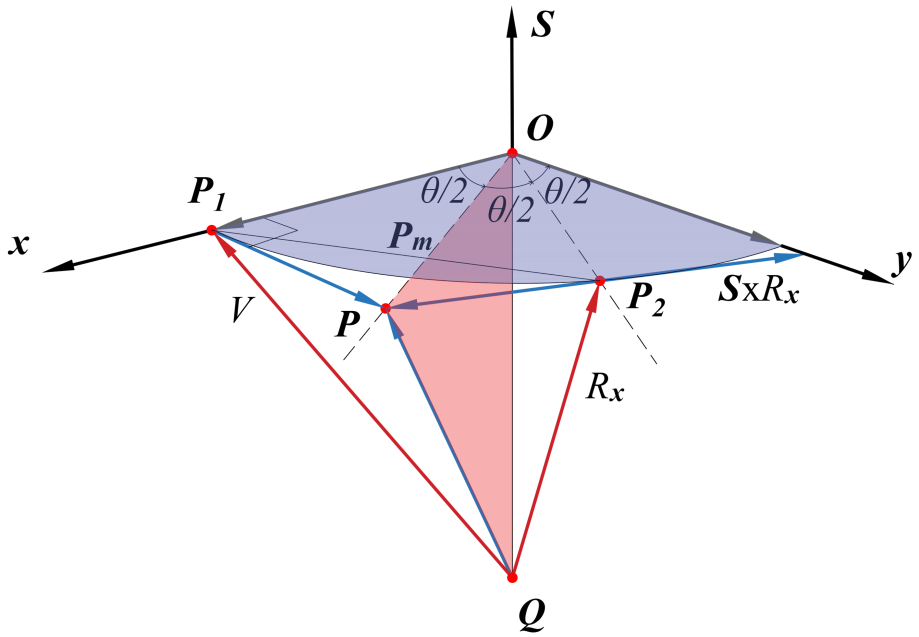


Figure 2.1 Geometrical derivation of the Euler–Rodrigues formula.

A skew-symmetric matrix $\boldsymbol{A}_s$ is determined by taking the vector cross product of $\boldsymbol{s}$ with a vector,

$$\boldsymbol{A}_s = [\boldsymbol{s}\times] = \begin{bmatrix} 0 & -s_z & s_y \\ s_z & 0 & -s_x \\ -s_y & s_x & 0 \end{bmatrix} \tag{2.4}$$

The Euler-Rodrigues rotation formula is constructed regarding to the property of skew-symmetric matrix $\boldsymbol{A}_s$ with $\boldsymbol{A}_s\boldsymbol{A}_s = \boldsymbol{s}\boldsymbol{s}^T = \boldsymbol{I}$ (the 3×3 identity matrix $\boldsymbol{I}$). The basic Euler-Rodrigues formula is as $\boldsymbol{R} = \boldsymbol{I} + \sin\theta[\boldsymbol{s}\times] + (1 - \cos\theta)\boldsymbol{s}\boldsymbol{s}^T$ [9]. The formula can be rewritten in a standard form as $\boldsymbol{R} = \boldsymbol{I} + \sin\theta\boldsymbol{A}_s + (1 - \cos\theta)\boldsymbol{A}_s\boldsymbol{A}_s$

The Euler-Rodrigues formula is used to construct a rotation matrix $\boldsymbol{R}$ as:

$$\boldsymbol{R} = \begin{bmatrix} {\omega_x}^2 + (1 - {\omega_x}^2)c & \omega_x\omega_y(1-c) - \omega_z s & \omega_x\omega_z(1-c) + \omega_y s \\ \omega_x\omega_y(1-c) + \omega_z s & {\omega_y}^2 + (1 - {\omega_y}^2)c & \omega_y\omega_z(1-c) - \omega_x s \\ \omega_x\omega_z(1-c) - \omega_y s & \omega_y\omega_z(1-c) + \omega_x s & {\omega_z}^2 + (1 - {\omega_z}^2)c \end{bmatrix} \tag{2.5}$$

where $s = \sin\theta$, $c = \cos\theta$.

In this thesis, the coordinates of the Platonic solids and truncated icosahedron before and after edge-rolling were represented by matrices $\boldsymbol{M}$ and $\boldsymbol{M'}$, where $\boldsymbol{M'} = \boldsymbol{M}\boldsymbol{R}$, respectively.

Chapter 3: Literature Review[1]

Computing collision-free paths, addressing clearance, and designing dynamic representations are examples of important problems with roots in computational geometry and discrete artificial intelligence search methods, and which are being revisited with innovative new perspectives from researchers in computer graphics, animation and robotics. Path planning of polyhedrons by rolling on edges includes arbitrary relocation, obstacle avoidance and goal achievement. These tasks have various applications in robotics fields such as autonomous vehicles, navigation and robotic surgery. This chapter reviewed path planning approaches applied to rolling contact with different types of polyhedral parts. The advantages and disadvantages of some mainstream algorithms were analyzed and compared. The chapter further pointed out the directions of future research on rolling polyhedron path planning in discrete space.

3.1 Introduction

Path planning algorithms pose one of the most challenging problems in nonholonomic systems to achieve the dexterous manipulation of objects in an unknown or partially known environment. This problem is mainly applied within the fields of robotics,

[1]This chapter is reproduced from the paper: N. T. Lam, I. Howard and L. Cui, "A Literature Review on Path Planning of Polyhedrons with Rolling Contact," 2019 4th International Conference on Control, Robotics and Cybernetics (CRC), 2019, pp. 145-151, doi: 10.1109/CRC.2019.00038.

artificial intelligent and autonomous vehicles. In robotics, the motivation of path planning is to find a possible path from an initial configuration, avoid the obstacles and achieve the goal configuration [10]. When assessing robot performance, there are two main kinds of planning: feasibility and optimality. The former is to find a plan for only achieving the path while the latter is to find an optimal path. In the artificial intelligence field, searching for actions to attain the desired goal state to receive the reward is employed including decision-theoretic methods. Each specific path planning algorithm is usually implemented in a parameter space such as a configuration space or a free space in which it generates a feasible path connecting the two given points. Defining the state space is also one of the important steps for planning purposes. The configuration space or C-space which includes all possible configurations in a physical system is applied for solving path planning problems in n dimensions. Examples of solving the path planning problems are presented by Lavelle [1] and Kavraki [11] showing feasible paths avoiding obstacles in high dimensional configuration space.

Rolling contacts between rigid bodies have been considered a nonholonomic system in order to solve the problem of dexterous manipulation of industrial parts. The goal of rolling manipulation is to roll the part from an initial configuration to a goal configuration. It can be divided into three types of rolling contacts: point contact [12, 13], line contact [14] and surface contact [15]. The simple experiment of a rolling polyhedral part on a table, mentioned in [16], showed that object manipulation with polyhedral surfaces without sliding can be executed by nonholonomic constraints through rolling. Some cases of rolling polyhedral objects through gras-

pless manipulation have been studied in the robotics field [17, 18]. Due to the lack of complete research on contact kinematics and rolling manipulation with discretized objects, planning for rolling polyhedral parts under reorientation with smooth and non-smooth systems still attracts attention from the research community.

With such path planning problems for polyhedra rolling contact as motivation, this chapter analyses and compares the different path planning algorithms and their application for rolling contact of polyhedra. The literature review is organized first and covers the properties of rolling polyhedra with the properties of polyhedra and rolling contact. The next section describes an overview of polyhedra and the two specific models - Platonic solids and Truncated icosahedron used in the study. The chapter then presents an overview of path planning algorithms and specifies two main path planning algorithms used for convex polyhedra rolling. Finally, the summary and gaps in the literature review will be presented.

3.2 Polyhedron Path Planning by Rolling Contact

Given a planar environment, the navigable space of the environment can be decomposed into cells such that the adjacency graph of the cells can be processed by discrete search methods. The process transforms the continuous path planning problem into a discrete graph search problem. For each node of the graph, there are only a discrete number of choices to decide in which direction to move next. While classical

differential geometry mainly focuses on smooth geometric shapes, discrete differential geometry investigates geometric shapes with a finite number of elements such as polyhedrons [19]. In terms of rolling contact in discrete space (the plane is discretized into different grids for each type of polyhedra), discrete paths include multiple curves which represent the rolling from a state to another state.

Multi-object or robot path planning in both simple and complex environments has received quite a lot of attention in the past while only a few studies have focused on discrete space. In particular, the combination of computation frameworks and discrete algorithms was considered in recent studies to capture the complex environment [20]. Previous studies of path planning for rolling contact between two rigid bodies focused on the contact of discrete surfaces to generate the path from an initial configuration to the final configuration [21, 22]. In this study, the polyhedral parts will be reviewed in the discrete path planning problems by changing their orientation.

3.2.1 3D Convex Polyhedra

This study mainly focuses on the path planning methods for convex polyhedral. It first introduces an overview of the $3D$ convex polyhedra. The geometric models are represented as polygonal objects, splines, or algebraic surfaces. They have different applications in computer-aided design and machining (CAD/CAM), robotics and

automation, manufacturing, computer graphics, animation and computer-simulated environments [23]. The $3D$ models are generally categorised into two main groups: non-polygonal models and polygonal models, as shown in Fig. 3.1.

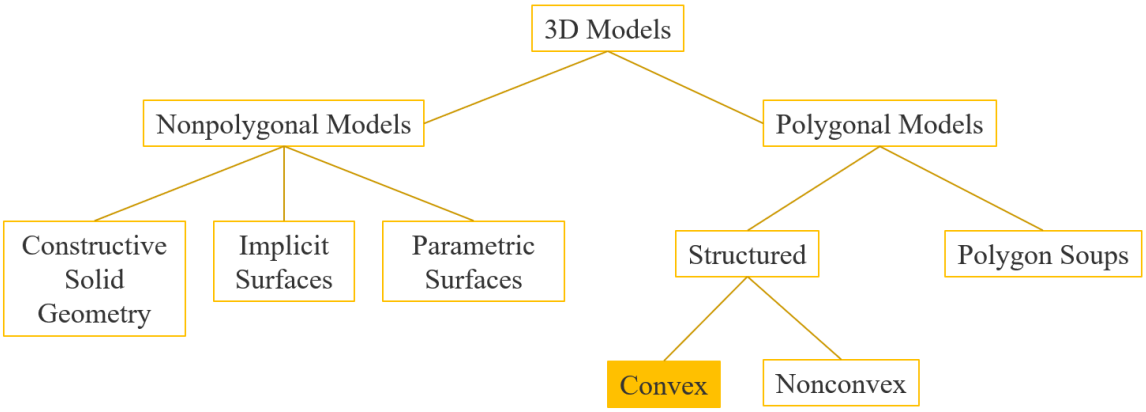


Figure 3.1 An overiew of $3D$ geometry models [23]

The first group of the $3D$ model is the non-polygonal models including constructive solid geometry (objects from primitives such as spheres, cones, or tories), implicit surfaces (the loci of points or called closed manifolds) and parametric surfaces (a description of surface boundary). In another hand, the second group is the polygonal models which are categorised into structured and polygon soups, which are the used models in modelling and computer graphic. The term "polygon soup" has evolved for describing arbitrary collections of polygons that carry no warranties concerning their structure. The structured models include convex and non-convex models. In this thesis, the path-finding algorithms through edge-rolling are applied to the convex polyhedra including the Platonic solids and the truncated icosahedron.

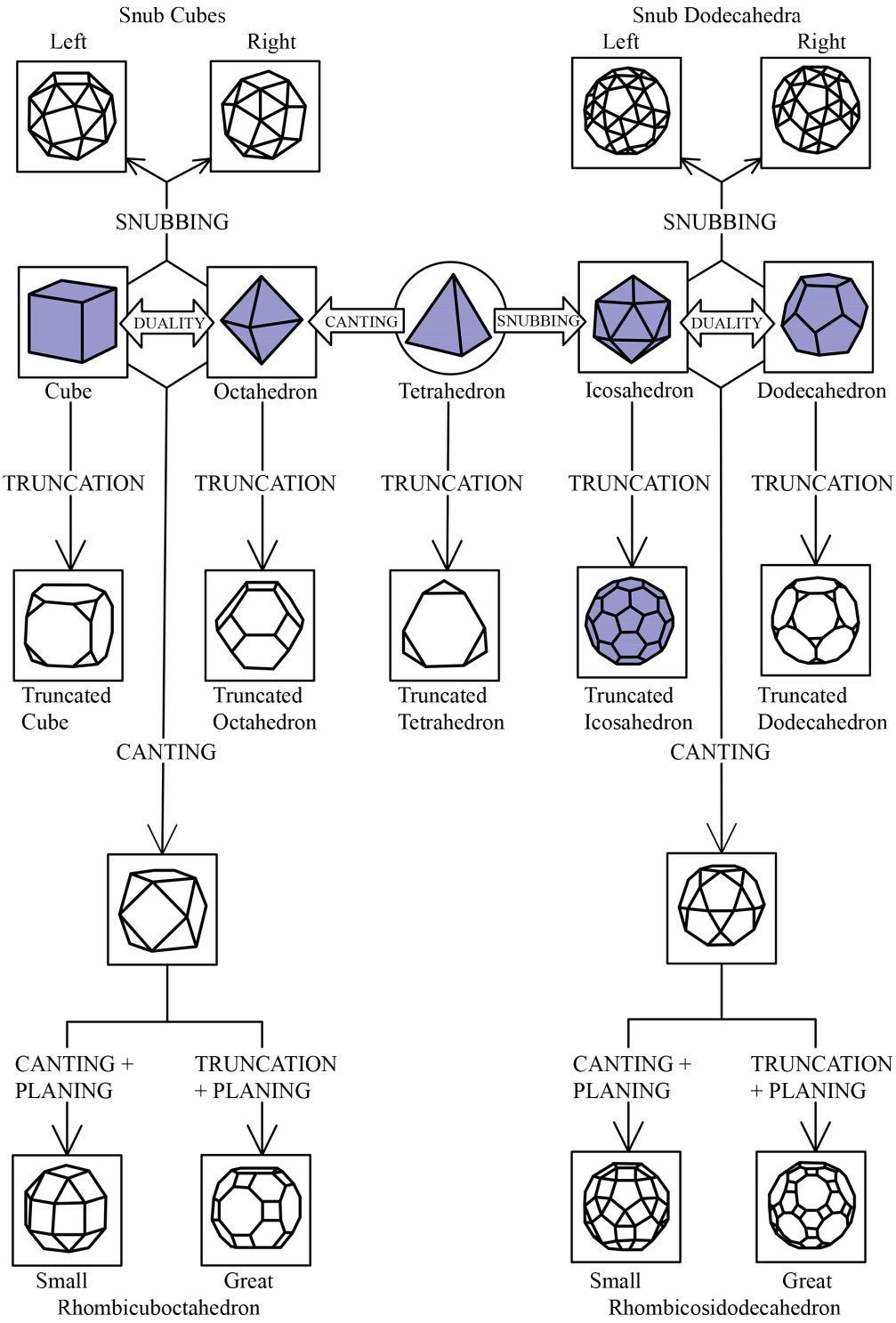


Figure 3.2 The Platonic and Archimedean solids [24].

Fig. 3.2 shows the relationship between convex polyhedra including the Platonic solids and the Archimedean polyhedra which are generated from the Platonic solids. It can be seen from the figure with the highlighted polyhedra, that the Platonic solids and Archimedean solids are completed regular or semi-regular with $3D$ shapes. Either the cube and octahedron (left) or the icosahedron and dodecahedron (right) can be derived from the tetrahedron (middle). Any truncated polygon is truncated from the Platonic solids. For example, the truncated icosahedron is truncated from the icosahedron or the truncated dodecahedron is truncated from the dodecahedron. The Archimedean solids have the same properties with regular length edges while having different faces. The truncated icosahedron is considered as an implementation of the path planning algorithm in this thesis. The literature reviews of the Platonic solids and the truncated icosahedron are detailed below.

3.2.2 Platonic Solids

The history of the Platonic solids —tetrahedron, cube, octahedron, dodecahedron, and icosahedron —can be traced back over 2000 years ago. In ancient Greece, Pythagoras (c.570-c.495 BC) knew of the tetrahedron, cube, and dodecahedron [25]. Plato (c.427-347 B.C), to whom the names of these five regular polyhedra are attributed, assigned them to the four basic elements —fire, air, water, and earth —as well as the heavens [26] (Fig. 3.3). In the 1600s, Kepler proposed a model of the solar system consisting of the Platonic solids set inside one another, distanced by

the inscribed and circumscribed spheres of each solid [27] (Fig. 3.4). The Platonic solids are all convex polyhedra bounded by a finite number of regular polygons. Being highly symmetric, Platonic solids have found many applications in mathematics, science, and art. For example, in studying molecules, they have been used to predict structure of crystals [28] or to reconstruct colloidal crystals from symmetric hard particles [29]. In mathematics, each Platonic solid was modelled on a $3D$ billiard table to demonstrate how a cube ball can move to hit every face and return to its starting point [30].

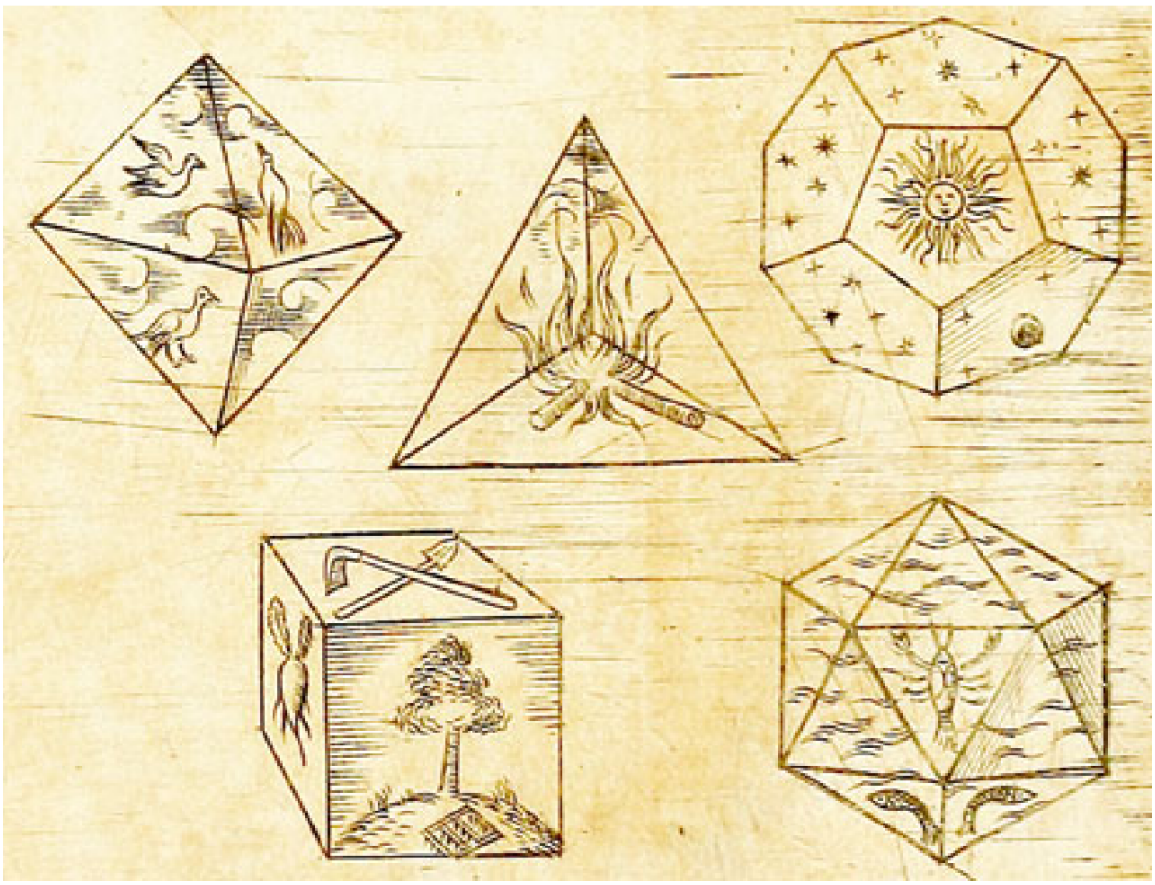


Figure 3.3 Platonic solids were represented as cosmic assignments by Kepler 1619 [31].

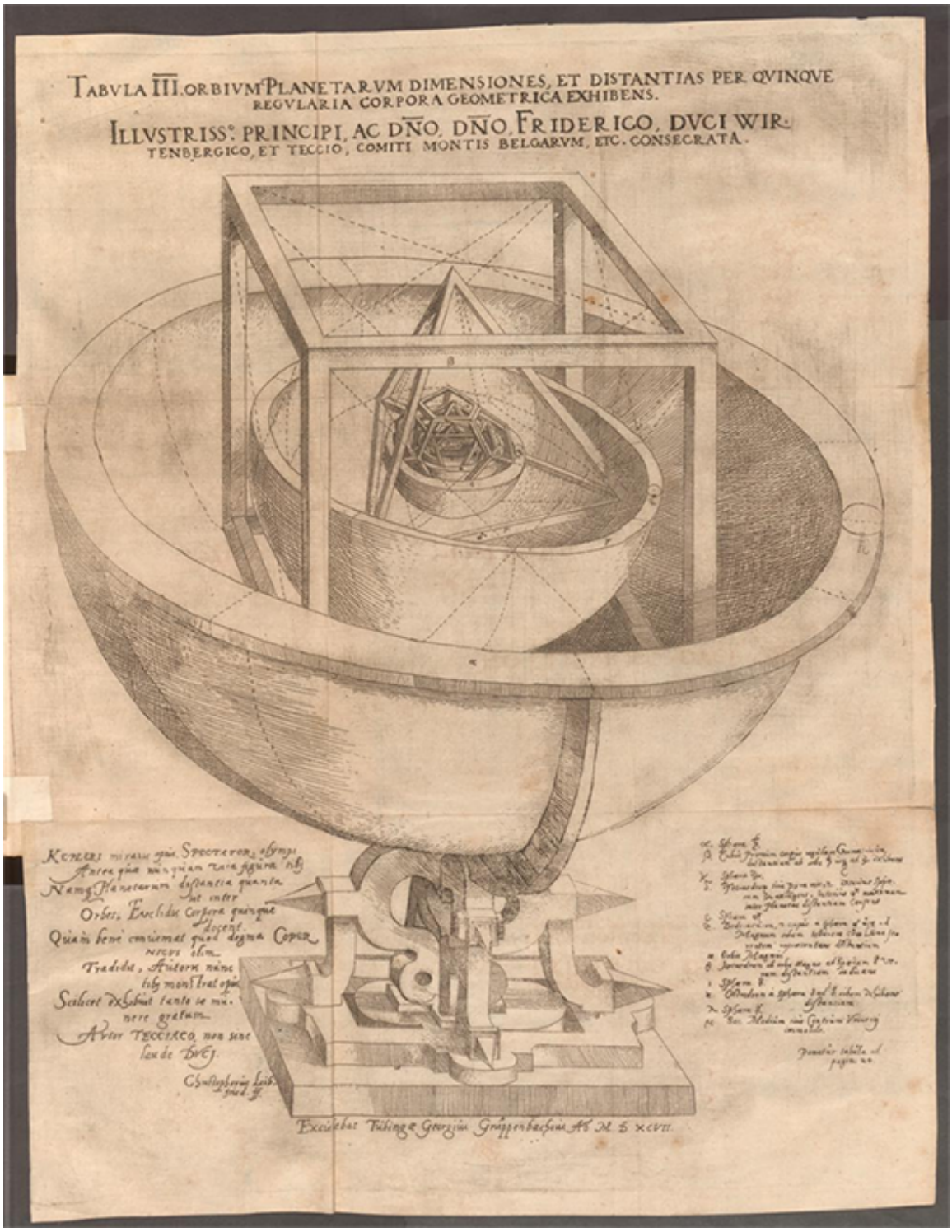


Figure 3.4 A model of the solar system consisting of the Platonic solids set inside one another, distanced by the inscribed and circumscribed spheres of each solid [32].

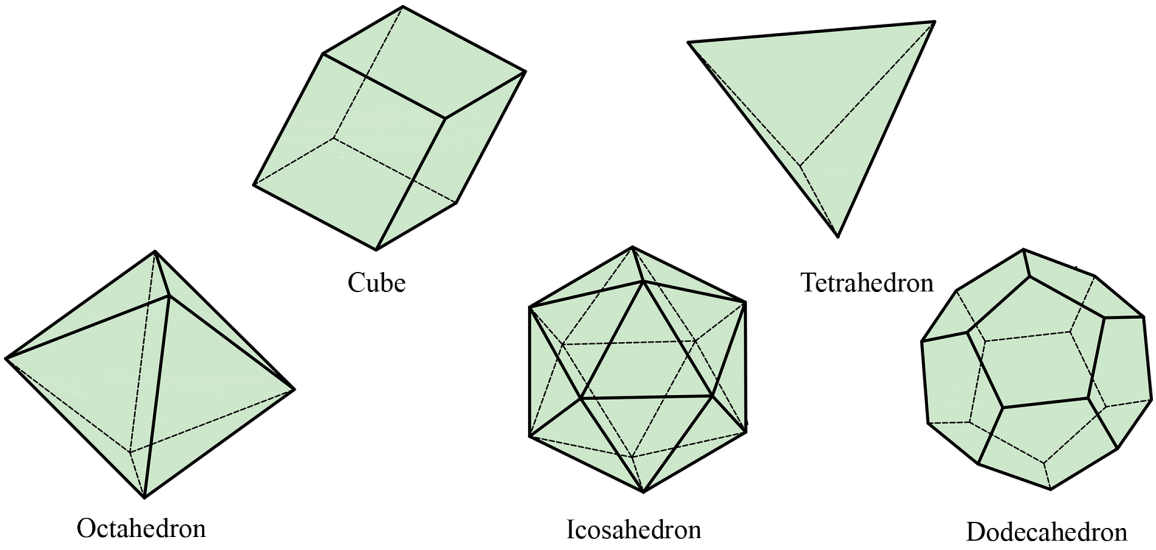


Figure 3.5 The Platonic solids consist of a cube, a tetrahedron, an octahedron, an icosahedron and a dodecahedron.

Properties of polyhedron

The Platonic solids, also called regular solids or regular polyhedra, are convex polyhedra with equivalent faces composed of congruent convex regular polygons. There are exactly five such solids: the cube, dodecahedron, icosahedron, octahedron, and tetrahedron, as was proved by Euclid. The $3D$ Platonic solids models are shown in the Fig. 3.5. The cube is constructed by 6 squares; the tetrahedron consists of 4 equilateral triangles joined at their edges into a triangular pyramid; the octahedron has a double-pyramid structure with 8 equilateral triangles; the icosahedron has 20 equilateral triangles; and the dodecahedron is composed of 12 regular pentagons (Table 3.1). The total number of vertices (V), edges (E), and faces (F) of the Platonic solids satisfy Euler's formula: $V - E + F = 2$ [33].

Table. 3.1 introduces the basic geometrical parameters of each Platonic solids with number of faces, vertices, edges, number of edges on each face and number of faces meeting at each vertex. These parameters will be initialised in the path planning algorithm for setting up the start-goal configuration of the polyhedra.

Table 3.1 Properties of polyhedron.

Platonic solids	**Faces** (F)	**Edges** (E)	**Vertices** (V)	**Edges on each face** (e_f)	**Faces meeting at each vertex** (f_v)
Tetrahedron	4	6	4	3	3
Cube	6	12	8	4	3
Octahedron	8	12	6	3	4
Icosahedron	20	30	12	3	5
Dodecahedron	12	30	20	5	3

Table. 3.2 shows the details of geometrical parameters of the Platonic solids consistings of the inradius (r_i), midradius (r_m), circumradius ($\mathbf{R}$) and rolling angles (θ). In this table, all the edges of the Platonic solids have the same unit length ($l = 1$). Fig. 3.7 shows an example of the rolling angle θ of the icosahedron. The angle can be calculated based on the dihedral angle ζ which is determined between two triangular surfaces with the common edge ($\theta = \pi - \zeta$). The Fig. 3.6 is an example to illustrate the dihedral angle between two surfaces. The line of intersection may be referred to as the edge of the dihedral angle.

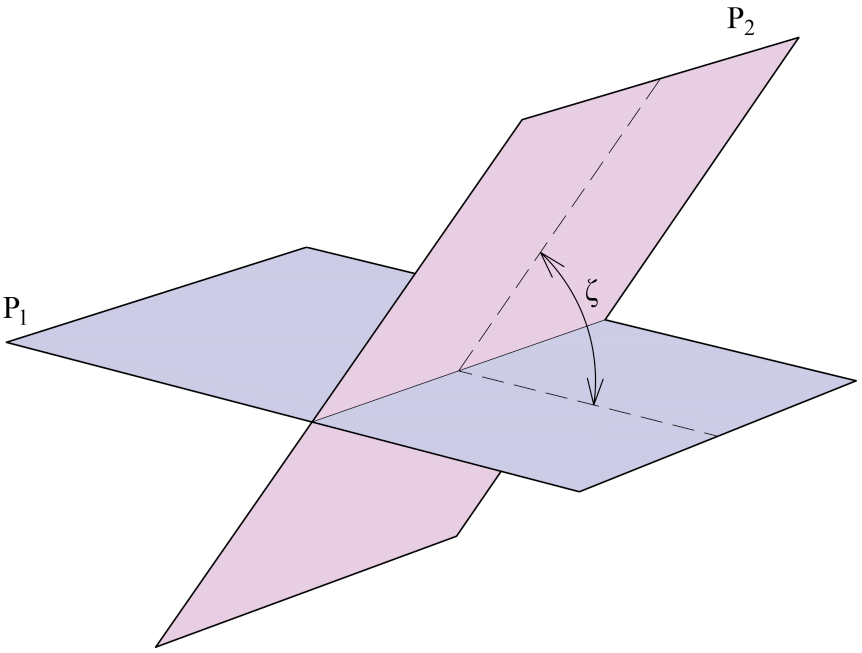


Figure 3.6 Dihedral angle ζ between two surfaces P_1 and P_2.

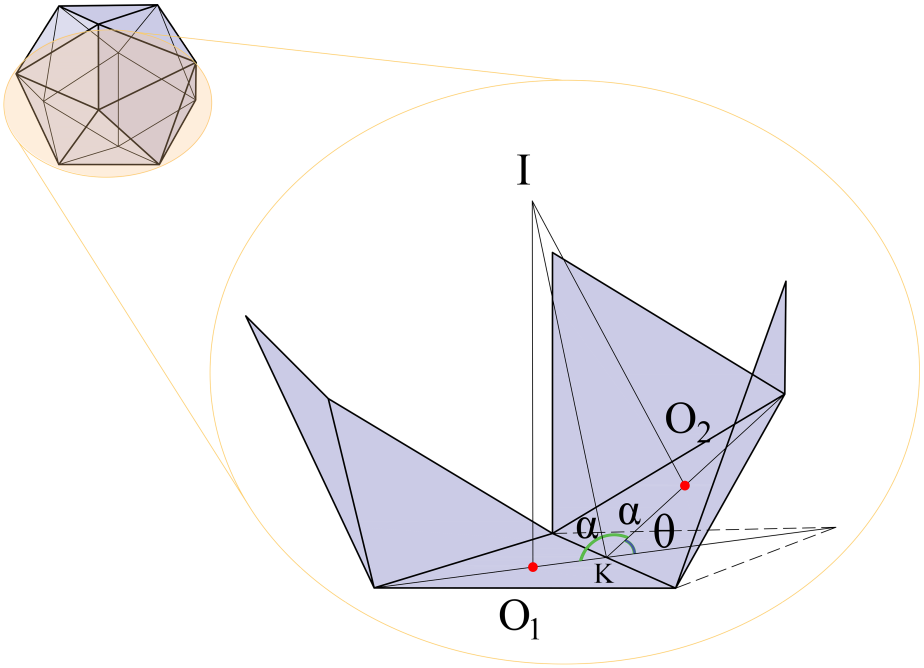


Figure 3.7 The rolling angle θ is calculated based on the dihedral angle $\zeta = 2\alpha$ of the icosahedron. In this illustration, I is the center of the icosahedron inscribed sphere. The current contact surface has the center O_1 while the next contact surface has the center O_2 ($IO_1 = IO_2$). The angle α is calculated based on the icosahedron midradius IK and the radius of the inscribed sphere of the iscosahedron IO_1 or IO_2.

Table 3.2 Geometrical parameters of the Platonic solids with inradius (r_i), midradius (r_m), circumradius ($\mathbf{R}$) and rolling angles (θ). In this table, all the edges of the Platonic solids have the same unit length ($l = 1$)

Platonic solids	r_i	r_m	$\mathbf{R}$	θ
Tetrahedron	$\frac{\sqrt{6}}{12}$	$\frac{\sqrt{2}}{4}$	$\frac{\sqrt{6}}{4}$	$2\arctan(\sqrt{2})$
Cube	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\pi}{2}$
Octahedron	$\frac{\sqrt{6}}{6}$	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\arccos(\frac{1}{3})$
Icosahedron	$\frac{1}{12}(3\sqrt{3}+\sqrt{15})$	$\frac{1}{4}(1+\sqrt{5})$	$\frac{1}{4}\sqrt{10+2\sqrt{5}}$	$\arccos(\frac{\sqrt{5}}{3})$
Dodecahedron	$\frac{1}{2}\sqrt{\frac{1}{10}(25+11\sqrt{5})}$	$\frac{1}{4}(3+\sqrt{5})$	$\frac{1}{4}(\sqrt{15}+\sqrt{3})$	$\arccos(\frac{\sqrt{5}}{5})$

Fig. 3.8 illustrates the geometrical parameters of the Platonic solids that are shown in the Table. 3.2. In this figure, O is the center of the model which is also the center of insphere (red), midsphere (green) and circumsphere (blue). I indicates the center of surface contact while K and V are the center of each edge and the vertex, respectively.

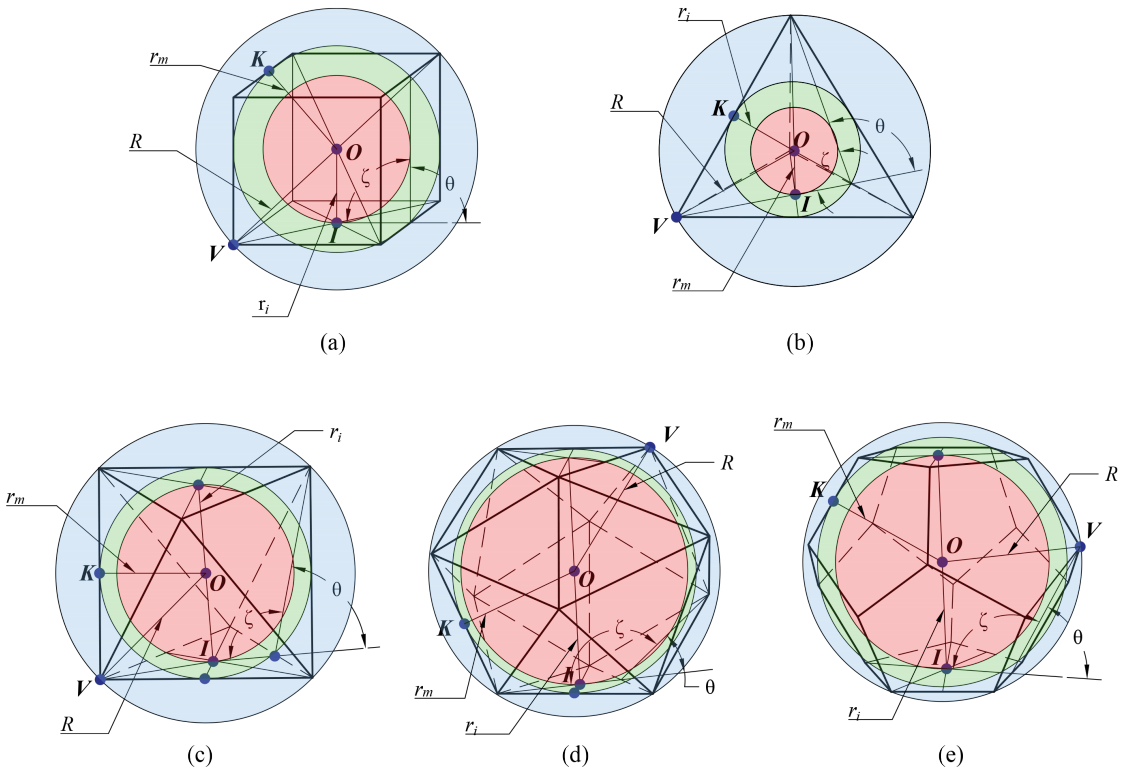


Figure 3.8 The illustration of the properties of the Platonic solids including the inradius r_i, midradius r_m, circumradius R, dihedral angle ζ and the rolling angle θ, which are shown in the Table 3.2. (a) Cube. (b) Tetrahedron. (c) Octahedron. (d) Icosahedron. (e) Dodecahedron.

3.2.3 Truncated Icosahedron

The truncated icosahedron is an Archimedean polyhedral truncated from icosahedron (one of the Platonic solids). A truncated icosahedron is the model represented in the construction of soccer balls with 32 polygons including 20 hexagons and 12 pentagons (Fig. 3.9).

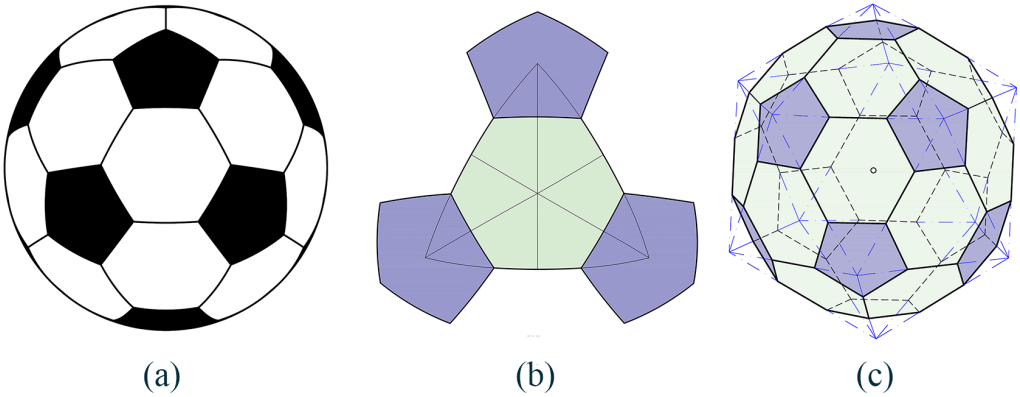


Figure 3.9 A soccer ball is based on the geometric shape of a truncated icosahedron.

Table 3.3 shows the general characteristics of the icosahedron with unit length $l = 1$. The golden ratio $\varphi = (1 + \sqrt{5})/2$ is a factor to represent other dimensions in the coordinate system. More details of the golden ratio property are shown in [34]. Path planning for the truncated icosahedron through edge-rolling considers the dihedral angles $\zeta_{F_{hexa}-F_{hexa}}$ (degree) between hexagon and hexagon faces or $\zeta_{F_{hexa}-F_{penta}}$ hexagon and pentagon faces to find the rolling angle $\theta = 180 - \zeta$. The two types of dihedral angles in the truncated icosahedron model are represented in Fig. 3.10.

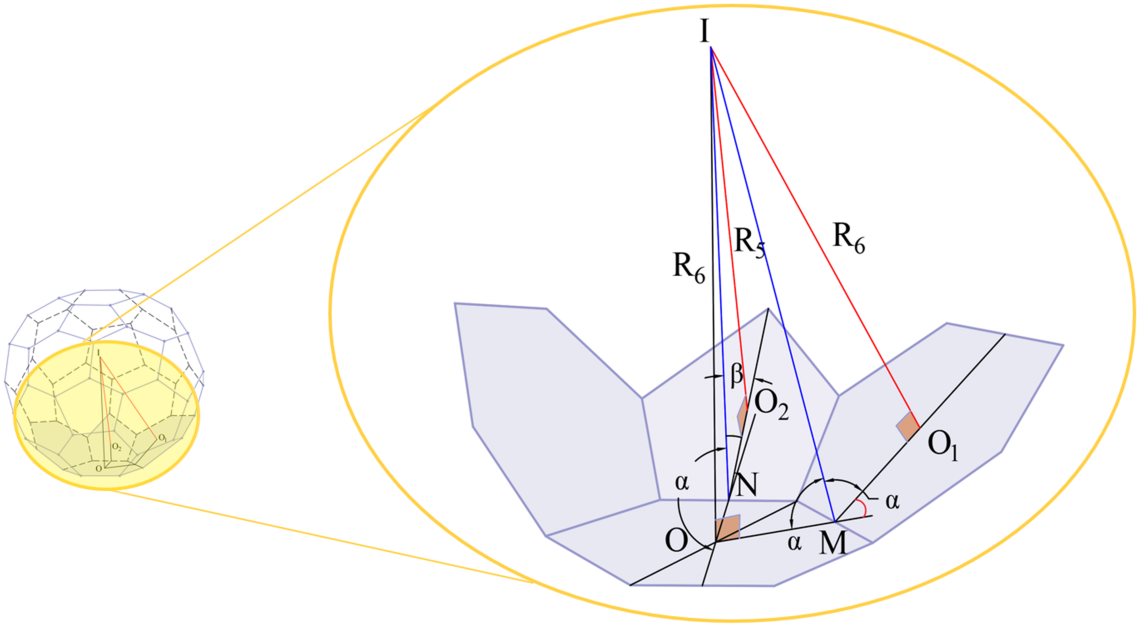


Figure 3.10 The dihedral angle between the hexagon face contact and the hexagon face is ($\zeta = 2\alpha$) while the dihedral angle between the hexagon face contact and the pentagon is ($\zeta = \alpha + \beta$). I is the centre of the inscribed sphere of the truncated icosahedron. The angle α is calculated based on the radius R_6 of the inscribed sphere to the hexagon faces and the radius of the inscribed circle of the hexagon face (OM). The angle β is calculated based on the radius R_5 of the inscribed sphere to the pentagon surfaces and the radius O_2N of the inscribed circle of the pentagon face.

Table 3.3 The geometrical properties of a truncated icosahedron with an unit length $l = 1$

Characteristics	**Unit length $l = 1$, Golden ratio $\varphi = \frac{1+\sqrt{5}}{2}$**
Cartesian coordinates	$(0, \pm 1, \pm 3\varphi), (\pm 1, \pm(2 + \varphi), \pm 2\varphi), (\pm\varphi, \pm 2, \pm(2\varphi + 1))$
Dimensions	Circumscribed sphere radius $r_{cs} = (1/4)\sqrt{58 + 18\sqrt{5}}$
Area	$\mathcal{A} = 15(2\sqrt{3} + \sqrt{(\sqrt{5} + 2)/\sqrt{5}})$
Volume	$\mathcal{V} = (1/4)\sqrt{125 + 43\sqrt{5}}$
Dihedral angles hexa_hexa	$\zeta_{F_{hexa}-F_{hexa}} = \arccos(-\sqrt{5}/3)$
Dihedral angles hexa_penta	$\zeta_{F_{hexa}-F_{penta}} = \arctan\sqrt{\frac{7+3\sqrt{5}}{2}} + \arctan\left((\tan(\frac{\pi}{5}))\sqrt{\frac{125+41\sqrt{5}}{10}}\right)$

3.2.4 Rolling Contact

Rolling contact has been studied in the fields of design and manufacturing, which mainly engaged in the knowledge of tribology [35, 36]. In this study, we briefly review the rolling contact theory in terms of geometry and its application in the robotics fields. The key idea of rolling contact is to maintain contact between two rigid bodies through a rolling motion under nonholonomic constraints. Rolling contact between multifingered robot hands and an object was studied in different research aspects such as geometry [37], controllability [38], motion planning [39] or robotic manipulation [7]. Due to the complex structure of the real multifingered robot hands with many degrees of freedom, the designing features of the robotic hand for rolling objects is a challenge for the purpose of dexterous object manipulation. Another work of rolling contact has been investigated with respect to in-hand manipulation, which is used for robotic grasping to manipulate objects with the tactile sensor [40]. In particular, such a simple end-effector through the robot hands can relocate only a few 3D shape objects, while rolling geometry between two surfaces of rigid bodies is more concerned with dexterity and manipulation capability.

Rolling Contact in Continuous Space

Rolling contact through ball-plate and rolling sphere problems of nonholonomic systems has been intensively studied in the past by many researchers [15, 41]. Hartmann

[42] applied a numerical blending method to develop the classical rolling ball method in terms of the constant and variable radius by analyzing the Voronoi surface, Bezier surface and G2-blending surface. The result of the study is achievable only by considering the parametric representation through numerical evaluation. Further, a rolling sphere model was proposed by Brockett [43] for the asymptotic stability problem of the five-dimensional nonholonomic systems that can be transformed into a chained form system. A theoretical idea of the path's optimization for nonholonomic manipulation was considered in this study for a kinematic problem of rolling rigid bodies. Another specific geometric formulation in terms of the curvature of rolling motion between a sphere and two arbitrarily shape fingers was derived by Montana [37]. The rolling contact condition has been formulated as a contact equation via the differential geometry concepts as a well-known nonholonomic constraint. This work successfully examined and analyzed the curvature form of point contact of an object while maintaining the rolling motion on a surface.

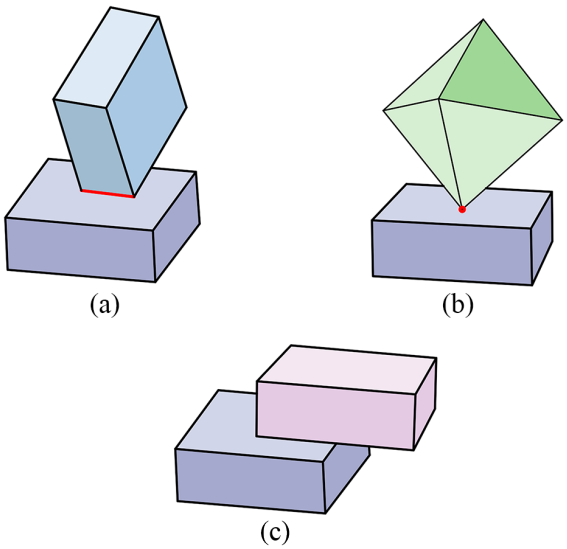


Figure 3.11 Main types of contact between two objects. (a) Edge-Suface contact. (b) Point-Surface contact. (c) Surface-surface contact.

Kinematic of Rolling Contact

A part of rolling contacts is considered in terms of the kinematics which are essentially analyzed in different fields such as dynamics [44, 45], controllability [46, 47] and motion planning [48, 49]. The majority of studies in multifingered robot hands have involved differential equations based on kinematics. Cai and Roth [12] used Taylor series expansion to derive the first and second order of kinematics of sliding-rolling. The paper focused on the relationship between the contact speed and the contact velocity which could be applicable to path planning tasks with motions of point contact. Okamura [50] improved Jacobian relationships to develop dexterous manipulation kinematics. The drawbacks of the system could be over-constrained or under-constrained, which hardly maintain the rolling contact property. Further, a series of studies about the kinematics of rolling contact was conducted by Cui [51, 52, 53, 54]. The author applied the theory of Darboux moving frames method from the differential geometry theory to demonstrate the contact equation between an object and multifingered robot hands and generate the forward and inverse kinematics of in-hand manipulation.

Contact Theory via Cartan's Moving Frame Method

Cartan's moving frame method has been essentially approached for geometric objects in contact kinematics [55, 56]. This method was widely applied in the computation

of symmetry groups and partial differential equations [57, 58], geometrical curves and surfaces [59] and finite-dimensional transformation groups from Lie algebra [60, 61]. Cui and Dai [62] used the moving frame and Darboux frame to derive the kinematics of rolling contact. This study was implemented in terms of curvatures of shapes through the spin motion to establish the contact theory. Another work [63] developed Cartan's moving method by reducing the order of derivation of three curvature invariants from the corresponding Frenet equations to analyze the deformation of a plane curve. These applications have been studied as a general theoretical foundation of moving frames for geometric properties of surfaces in Euclidean space and Riemannian geometry, which has not much attention to the robotic field.

Planning Motions of Polyhedra

Path planning for polyhedra by edge-rolling is scarce. Some results that have been obtained in the study of the manipulation of objects by rolling, in the view of the realization of a robot gripper that exploits rolling to achieve dexterity, such as the ability to arbitrarily change the location and orientation of the manipulated objects. An early prototype of the such a device was presented by [64], along with some preliminary experiments in planning and controlling motions of a sphere manipulated by rolling. Marigo *et al.* [65] applied manipulation by rolling to objects of polyhedral shape. The design of grippers exploiting rolling was based on the conjecture that a kinematic system comprised of almost any pair of rolling surfaces is controllable,

which has been shown true lately by [38].

Formulas for predicting how the contact points and the relative orientation of the surfaces evolve with rolling have been investigated first by Cai and Roth [13] and Montana [37], independently. Early work on this subject has been done by Cole *et al.* [66], and Li and Canny [67], who studied the problem of rolling by putting it in the framework of nonlinear control systems theory, and showed that a ball rolling on a plane can be displaced and reoriented at will within its five-dimensional configuration manifold by only using two inputs. A geometric algorithm was proposed by these authors to plan the motions of a very particular case (a sphere rolling on a plane).

Octahedron edge-rolling was firstly introduced in [68] to tackle the problem of manipulating polyhedral parts from a more application-oriented viewpoint. The algorithm is to provide an algorithm for planning such manipulation by giving an initial and final configuration pair, to find a sequence of simple rolling motions that brings the part from the former to the latter. However, the completed path planning failed due to errors propagating and needs a translation step to achieve to goal.

Erdmann *et al.* [18] proposed a planner that determines a sequence of tiling operations designed to mminimize the uncertainty in the orientation of the orientation of the polyhedron. The shape is limitted to expend to a general polyhedra. Only rolling-cube puzzles were solved by using NP-hardess Hamiltonian path in grid graphs [69].

Summary

The literature shows that manipulation polyhedra tasks considered both changing position and orientation. The task of rolling between rigid bodies in three-dimensional space is a well-known case of nonholonomically constrained motion. The plate-ball system or in-hand object manipulation is the most of the applications. The convex polyhedra rolling is still challenging such as path-finding for the Platonic solids to achieve the target pose on a surface. Only rolling-cube puzzles were considered but the approach may not extend to general polyhedra.

It is evident that there currently does not exist a state-of-art technique to solve rolling polyhedra tasks of achieving the goal configuration. More complex geometrical polyhedra will increase the constraint in the path planning problems. The next section considers path planning methods to apply for the polyhedra rolling task and surveys on how to deal with the literature gaps.

3.3 Path Planning Algorithms

Path planning has been the most important task of robotics research in both static and dynamic environments as an emerging area for a long time [10, 70]. The path planning strategy for robotic research can be categorized into traditional methods and AI-based approaches or also divided into two folds such as the local path planning

and the global path planning strategy. Most of the previous studies focus on mobile robots [10], unmanned aerial vehicles (UAV) [71] or autonomous self-driving cars [72] while only a few pieces of research have been implemented on the rolling contact of robotic in-hand manipulation. This section will present an overview of path planning algorithms in terms of traditional and AI-based planning approaches as shown in Fig. 3.12.

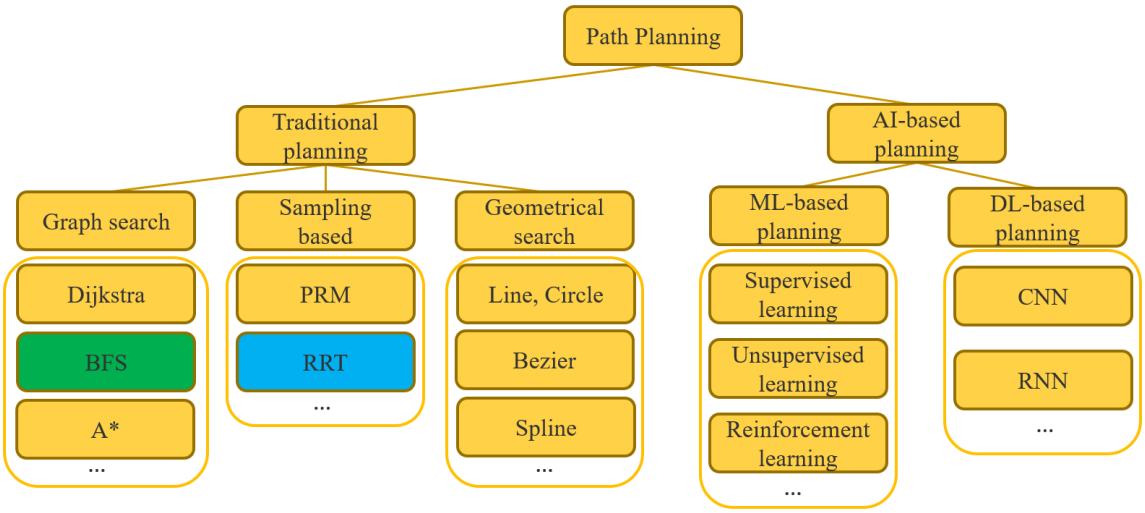


Figure 3.12 The path planning algorithms can be divided into traditional planning and AI-based planning. Traditional path planning includes graph search, sampling-based and geometrical search while AI-based path planning consists of machine learning-based path planning and deep learning planning.

3.3.1 Traditional Path Planning

The basic idea of path planning in most cases is that state-space models will be used to demonstrate the distinct situation in which the task of a planning algorithm solves the sequence of actions transforming from an initial state to other states. For example, Thomas [73] applied Delaunay triangulations to discretize the environment, and cubic spline representations are proposed to meet robot kinematic constraints.

Considering the continuous curvature on smooth curves has been integrated within the probabilistic approaches in order to compute the piecewise smooth paths for a car-like vehicle as a four-dimensional system [74]. Whereas when dealing with nonholonomic constraints, a sampling-based road map technique has been proposed in [75] which determined trajectories and re-entry trajectories for hovercrafts and rigid spacecrafts. Based on decomposing space into cells [76], a potential field without local minima was assigned with polygonal partitions of planar environments to solve Laplace's equation problems in each presence cell. Applications for these techniques in discrete space are limited by a grid.

Graph-search Algorithm

Graph-search-based algorithms can be divided into the depth-first search, breadth-first search, and best-first search [77]. The depth-first search algorithm builds a search tree as deep and fast as possible from origin to destination until a proper path is found. The breadth-first search algorithm shares similarities with the depth-first search algorithm by building a search tree. The search tree in the breadth-first search algorithm, however, is accomplished by extending the tree as broad and quick as possible until a proper path is found. The best-first search algorithm adds a numerical criterion (value or cost) to each node and edge in the search tree. According to that, the search process is guided by the calculation of values in the search tree to decide: (1) whether the search tree should be expanded; (2) which branch in the

search tree should be extended. The process of building search trees repeats until a proper path is found. Graph search algorithms are composed of many algorithms.

Randomized Potential Field Algorithm

Based on the potential function, randomized potential field methods have been proposed and implemented for the classical mover's problems, which can overcome the local minima with low computational complexity [78]. Pre-computation of a connectivity graph of the global path planning which contains the guide for grid search in the configuration space is the high cost for the computation system to escape the minima at the goal configuration. The technique from the first step is the best-first search which does not require a local minimum of the potential function to be reached in high-dimensional configuration spaces. Then the search algorithm proceeds along the negated gradient of the potential function until achieving the goal configuration. The most powerful aspect of this method is the discretization of the configuration space and the workspace into a hierarchical bitmap grid that can be applied for many degrees of freedom of robots.

Dijkstra's Algorithm

The aim of the Dijkstra method is to determine the shortest path in a graph based on a known source of vertices [77]. The basic Dijkstra algorithm is a breadth-first-search which includes storing all the vertices (nodes) at a first step with a distance of "infinity", then assigning a distance value from the first node to the other nodes.

Next, all the connected nodes are updated with the new distances to calculate the distances, then taking the comparison between these distances and then choosing the shortest path from the initial node. This algorithm faces the issue of larger memory of the storage structure which can lead to limited applications for indoor environments in terms of autonomous robot path planning [20].

Heuristic Search Method

The fundamental robotic path planning problem is to represent the environment as a graph involving the set of possible robot locations and a set of edges that can generate the path. The popular method for determining the least-cost paths is A* as the Heuristic-based search algorithm in [79]. The search algorithm must expand the fewest possible nodes in order to make searching for an admissible path. Then the evaluation of available nodes is needed to determine the next efficient nodes. The initial search approached by the A$\star$ algorithm takes two steps to generate an optimal path by receiving information from one of the initial cells in free space and re-planning from scratch when the environment has changed to expand a new cell. However, the A* computation process needs high configuration processors to successfully reach various nodes. In the real-world scenario, the search operation may sometimes be performed with inaccurate planning graphs.

D-Star (D*)

The D* algorithm [80] based on dynamic A* is applied to the mobile robot path planning in the real-time path calculation and the optimization of replanning to achieve the goal robot's location from its current location. Like the A* technique, D* can use the heuristic search to find the direction of the robot and optimize the cost of state expansions which can be used to generate the shortest path from the current robot's location to other locations. To compare with the Dijkstra algorithm, D* takes advantage of the previous search calculations incrementally requiring recomputation of all calculation steps in the Dijkstra algorithm.

Theta-Star (Theta*) and Lazy-Theta-Star (Lazy Theta*)

Theta* and its extensions - Lazy Theta* (Any-angle path planning algorithm) [81, 82] are the famous path planning techniques on grids based on A* with the much shorter path by propagating angle ranges. These techniques are applied to solve the shortest graph problem implemented on the grid including block and unblock cells where block cells can be seen as obstacles. The basic algorithm is to find the shortest path from a start vertex to the goal vertex where a continuous environment has been discretized into a grid. In one cell, the shortest distance is represented by the diagonal line. The shortest path called a visibility graph is connected from the goal vertex to one of the vertices of block cells and to the start vertex. The drawback of the Theta* algorithm is that it is not guaranteed to determine the shortest path in the continuous environment.

Breadth-First Search Path Planning

In this work, the BFS and RRT path planning are used to find the path for convex polyhedra through edge-rolling. BFS is an algorithm used for traversing graphs or the tree expansion which means visiting each node of the graph. It is of essential attention to operations research [83, 84, 85, 86] and has wide application in computer science [87], robotic [88, 89] and automation fields [90]. The BFS is a recursive algorithm to search all the vertices of a tree (Fig. 3.13). Once the algorithm visits and marks the starting node, then it moves toward the nearest unvisited nodes and analyses them. All nodes are marked when they are visited. These iterations continue until all the nodes of the tree have been successfully visited and marked.

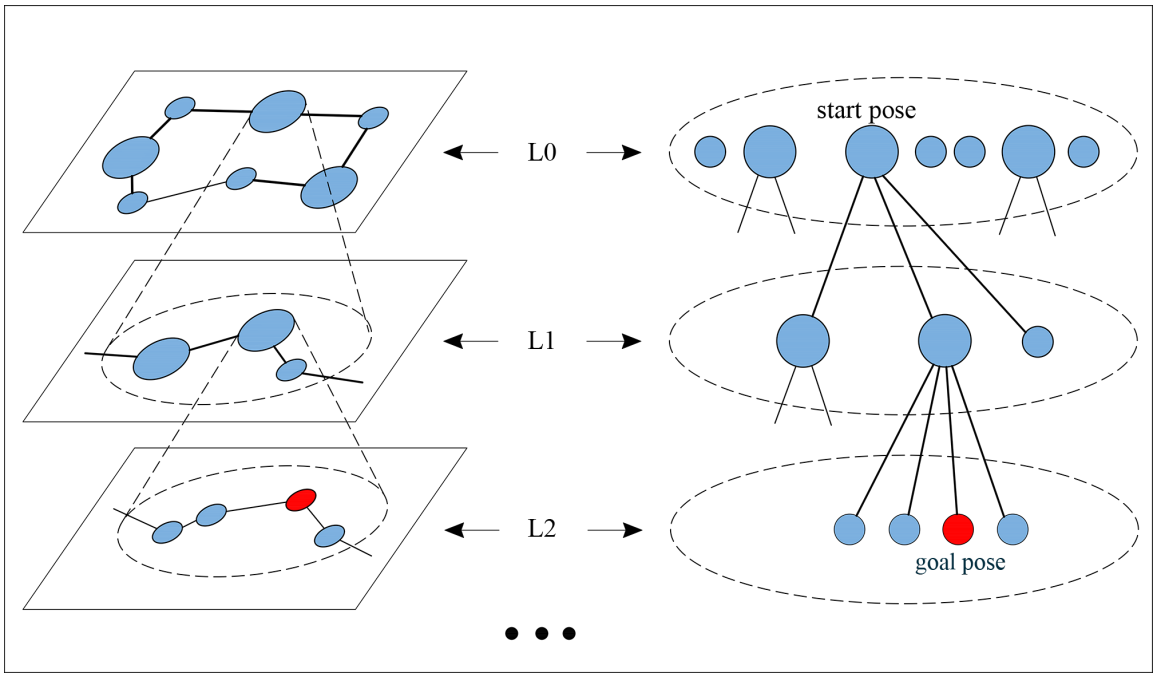


Figure 3.13 BFS path planning with search tree. Starting form a node (pose of any Platonic solid), the tree expands from the root until the leave achived the goal pose (the red state).

The BFS technique has several practical applications with the large network analysing. Ahn *et al.* [91] used BFS to analyse a complete online social network Cyword and validate snowball sampling method to obtain partial network topologies of Orkut and MySpace. Mislove *et al.* [92, 93] studied BFS to examine multiple online social networks Orkut, Flickr, LiveJournal and Youtube. The study discussed the core of matching indegree of user nodes and outdegree, and the links between small clustered groups and low-degree nodes. Wilson *et al.* [94] studied the user interaction graph in the Facebook social network using several BFSs, each BFS constrained in one of the largest 22 regional Facebook networks. Achlioptas *et al.* used this searching idea in [95] to study the bias of trace route sampling in random graphs with a given degree distribution. The basic operation in [95] is trace route ("discover a path") and is performed from a single node to all other nodes in the graph.

The BFS algorithm is used for path-finding in the application of rolling Platonic solids due to various reasons. There are some important aspects of the algorithm that make it first considered for the path planning of the Platonic solids. BFS algorithm has a simple and robust architecture for searching for the goal in configuration space. All the nodes of the tree are visited and the shortest path through the nodes is generated by the BFS algorithm. An advantage of the BFS is that the paths can be generated with a high level of accuracy. Traversing through the tree graph with the lowest number of iterations (in limited search space) which are continuous and avoiding an infinite loop issue is also another advantage of the technique.

Sampling-based Algorithms

Sampling-based algorithms randomly sample a fixed workspace to generate sub-optimal paths. The rapidly-exploring random tree (RRT) and the probabilistic roadmap method (PRM) are two algorithms that are commonly utilized in motion planning.

Probabilistic Roadmap Planner (PRM)

The PRM technique has been applied for motion planning problems to generate a path for robots without colliding obstacles from the environment over the past two decades [96, 97]. These early studies were only implemented on the 2D workspace and holonomic motion planning. The PRM algorithm is normally used in a static scenario [11]. The PRM computation consists of two phases: the preprocessing phase and the query phase. Repeating the step of generating random free configuration space can generate a probabilistic roadmap in the preprocessing phase. The nodes of the graph and the paths are computed through a local planner that can achieve the graph edges. In the query phase, it starts with the connection between the initial and the goal configuration by Dijkstra's shortest path query [98]. A feasible path can be extracted from a graph search by finding complete edges from the connecting nodes in the roadmap. Since the PRM has these properties, this method should be optimized due to the low quality of the searching process - the graph is a tree, not cycle graphs - and the straight-line motion which generates the first-order discontinuities at the nodes.

Rapidly-exploring Random Trees (RRTs)

The RRT algorithm is more popular and widely used for commercial and industrial purposes. It constructs a tree that attempts to explore the workspace rapidly and uniformly via a random search [99]. The RRT algorithm can consider nonholonomic constraints, such as the maximum turning radius and momentum of the vehicle [100]. Rapidly-exploring random trees (RRT) [101, 99] is a fast probability planning that can be extended to multidimensional spaces (Fig. 3.14). More specifically, the RRT is probabilistically complete [102] so that it guarantees the complete discovery and exploration of the map. The RRT algorithm extends a tree structure from the starting point and follows an expansion direction, as determined through random selection points in the planning space. Based on the RRT algorithm, an opportunity-constrained sub-path can be generated from the starting position to the target [103]. The RRT path planning algorithm can be utilized to detect frontier points in the developing map. Umari *et al.* [104] proposed the idea of using the RRT to detect frontier points. Furthermore, since the RRT algorithm is biased towards the undetected area, it can quickly detect the frontier points of the map.

RRT is also defined as a randomized data structure technique for solving planning problems [105]. This sampling-based algorithm does not require any connections of nearby configurations, which can be applied for path planning problems under holonomic, nonholonomic and kinodynamic constraints. A key advantage of the RRT technique is to handle high degrees of freedom problems for robotics but the method

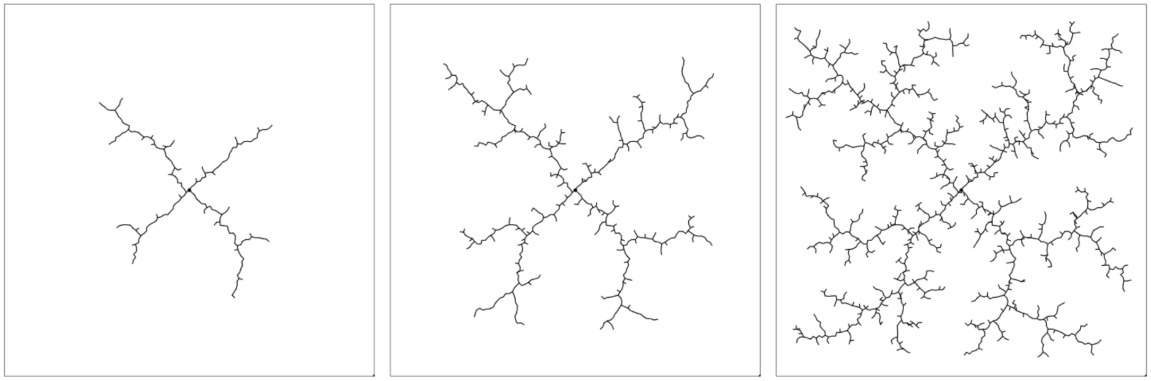


Figure 3.14 The RRT quickly expands its nodes in a few directions to randomly explore the four corners of the square [105].

still remains on the trajectory optimization problems. Many algorithms [106, 107, 108] that extend the RRT algorithm have been studied. The study in [109] improved the RRTs method called the RRT-Connect technique which combines the RRTs and a greedy heuristic method to speed up the exploration of configuration (state) space and the connection from an initial configuration to other configurations.

The above algorithms show more efficient performance by improving the RRT algorithm to overcome the limitations of sampling-based methods but they are still not perfect. Their limitations include being unable to derive the optimal length and there is room for improvement in terms of the number of operations and time. For example, the RRT* algorithm has rewiring (search for the parent node as a via point nearby a newly inserted node, where the addition of path length from the start point to the via point and path length from the via point to the newly inserted node in the tree is the optimized, and change the neighbouring nodes to optimize the path length) and neighbour search (search for nodes nearby the node to be newly inserted in the tree) processes to obtain shorter path lengths than the RRT algorithm [102].

However, there is an efficiency trade-off in this process. In other words, while the convergence rate has improved, the planning time has significantly increased [110]. Therefore, the RRT* algorithm cannot be said to be better than the RRT algorithm in all performance metrics and it can be said that the RRT* algorithm gets closer to the optimum at the expense of planning time.

Geometrical Search Algorithms

Interpolating curve algorithm is defined as a process that constructs or inserts a set of mathematical rules to draw trajectories. The interpolating curve algorithm is based on techniques (e.g. computer-aided geometric design (CAGD)) to draw a smooth path. Mathematical rules are used for path smoothing and curve generation. Typical path smoothing and curve generation rules include line and circle [111], clothoid curves [112], polynomial curves [113], Bezier curves and spline curves [114]. However, the path-finding through interpolation curve has a high cost to reach the goal although generating accurate paths.

3.3.2 Artificial Intelligence Approach

Artificial Neural Network (ANN)

ANN is a mathematical or computational model based on the function of a biological neural network. Various applications of ANN have been proposed in classification,

data compression, pattern recognition or prediction. Neural network-based path planning for mobile robots has been widely implemented in different environments with/without obstacles [115, 116]. The main purpose of the technique is to find the path autonomously from the dynamic activity landscape of the neural network. It can predict the feasible paths with obstacle avoidance which makes the robot achieve the goal of the environment. However, the training process requires data input and computational costs due to a large number of training samples.

Machine Learning (ML)

Machine learning is a field of computer science that is applied to various problem domains which required data. Many of these domains involve training a system to perform a task or finding the optimal solution to a complex problem. A full survey of machine learning methods is beyond the scope of this thesis, but a brief introduction of a few machine learning subfields are presented in particular relevance to specific algorithms that have recently been applied to path-planning and/or the environmental representations used for path-planning. Supervised learning algorithms teach a system to discriminate between various groups or classes of input, based on examples of each class. The input is usually represented as a vector, where each dimension of the vector corresponds to a particular attribute or feature[117]. Supervised regression is similar to the idea described above, except that the algorithm attempts to map feature vectors to real number values instead of non-ordinal class labels. Regression

is useful when output is used as a measure [118]. The main applications of the ML path planning algorithms are in robotic systems which have many parameters that need to be turned [119].

Reinforcement Learning (RL)

RL is a machine learning framework based on trial and error which is applied in discrete state spaces [120]. One of the most popular RL algorithms is Q-learning widely used in mobile robot path planning [121]. The learning process of Q-learning consists of determining the goal of the agent on the map by receiving the maximum sum of rewards from the environment. With continuous and very large state spaces which lead to a high cost of going from an initial state to the current state, the Q-learning algorithm will be expensive for the computation process. Path planning is realized by attaching destination and safe paths with big rewards (numerical value), while obstacles are attached with penalties (negative reward). The optimal path is found according to total rewards from the initial place to the destination. To better understand optimal value RL, it is necessary to recall several fundamental concepts: Markov chain, Markov decision process (MDP), model-based dynamic programming, model-free RL, Monte-Carlo method (MC), temporal difference method (TD), and State-action-reward-state-action (SARSA). MDP is based on Markov chain [121], and it can be divided into two categories: model-based dynamic programming and model-free RL. Mode-free RL can be divided into MC and TD which include SARSA and

Q-learning algorithms. All the DL techniques can be used in path planning problems. However, it is pointed out that DL requires vast amounts of hardware and energy to train data.

3.3.3 Summary and Gaps

The literature provided an aspect of discrete path planning for the polyhedra parts through rolling contact. Rolling contact between two objects under nonholonomic constraints is a difficult task to work with. In terms of path planning under rolling constraints, it is necessary to understand the problem of nonholonomic systems and consider geometric aspects of the complex polyhedral systems.

The literature has categorized path planning algorithms into two categories. The first category is the traditional path planning algorithms, which are more accessible to solve the path planning related to limited constraints and simple environments. Although their drawbacks include high computation time and intensive memory for the planning processes, the outcomes of the algorithm can guarantee finding feasible paths. The second category AI-based approaches, which provide efficient and fast convergence to generate the shortest path. Although each technique has some disadvantages such as training time or limited hardware supports, these approaches have been applied successfully to many complex path planning tasks in terms of the global optimal path, uncertain environment and online real-time path planning.

Discrete path planning has the potential to be a technique for convex polyhedra through edge-rolling. The BFS and RRT algorithms can solve the path-finding problem for rolling-polyhedra. However, based on the literature review, the completed path planning for polyhedra with rolling contact has not been solved. The following comments express those gaps:

- First, manipulation polyhedra which are considered a nonholonomic system, as it can be checked by rolling a die on a table along cyclic paths, needs a completely different set of tools to plan rolling motions of the polyhedra without considering the orientation [6].
- Second, it is unclear to generate the path for rolling polyhedra on a plane [18]. Furthermore, the polyhedra part rolling is solved by reorienting and displacing to achieve the goal [68] while the path planning problem without object translation has not yet been addressed.
- Third, the literature does not provide any path planning algorithms to find the edge-rolling path for general polyhedra to achieve the goal configuration while avoiding obstacles.

Chapter 4: BFS and RRT Search for Path Planning Algorithms

Path planning for rolling between rigid bodies in space is as well-known non-holonomic system. The process of finding a feasible and optimum global path for rolling-polyhedra usually involves determining the shortest discrete path, which will be subsequently smoothed to satisfy the requirements of reaching the goal configuration. Within this context, the first part of this chapter reviews the literature on the BFS algorithm. Based on its performance with graph searching, the chapter then provides the potential path planning methods to find the shortest path for rolling objects. Due to some limitations of the BFS algorithm such as the complexity and the computation time, the BFS-based algorithm is only proposed to solve the path planning problem for the special convex-polyhedra - the Platonic solid. In the second part, the planning algorithm by using the RRT algorithm to find rolling-paths for convex polyhedra is proposed. The algorithm is also a kind of probabilistically complete exploration algorithm based on the tree expansion structure. A review of related works highlights the shortcomings in the literature. The notation and problem statement of the basic RRT and the configuration space for rolling through edge-contact using RRT algorithm are then detailed. The details of the path planning based on RRT method for convex polyhedra through edge-rolling to achieve the closest goal configuration are then explained. Finally, the chapter highlighted the potential interest

of the proposed methods for successfully finding rolling paths for convex polyhedra through edge-rolling on a plane.

4.1 Configuration Space

The path planning problem is the task of finding a collision-free path for a robot around multiple obstacles. It is also referred to by its classical name, Piano movers problem. Consider an example of a moving object on a plane and the object needs to get to the goal configuration from a start position. If there is a series of rotations of the object that moves it from the starting pose to the goal pose without colliding with any obstacles, then there exists a path for the object to reach the goal.

A configuration of the object is a specification of the positions of all the points of the object relative to a fixed coordinate system. The configuration space of the object with respect to the obstacles around it, is the set of all possible configurations of the object and we denote it by $\mathcal{W}_{space}$. A subset of $\mathcal{W}_{space}$ with all configurations where the object and the obstacles do not overlap is called $FreeSpace$ and denoted by $\mathcal{W}_{free}$. The blocked space is the set of configuration objects there is an overlap, we denote it by $\mathcal{O}_{obs}$. The $\mathcal{W}_{free}$ and the $\mathcal{O}_{obs}$ are both open sets, the boundary between these sets is called the $ContactSpace$. $\mathcal{O}_{ct}$ is a subset of $\mathcal{W}_{free}$, which is the space of all configurations where the objects have contact with the plane (4.1).

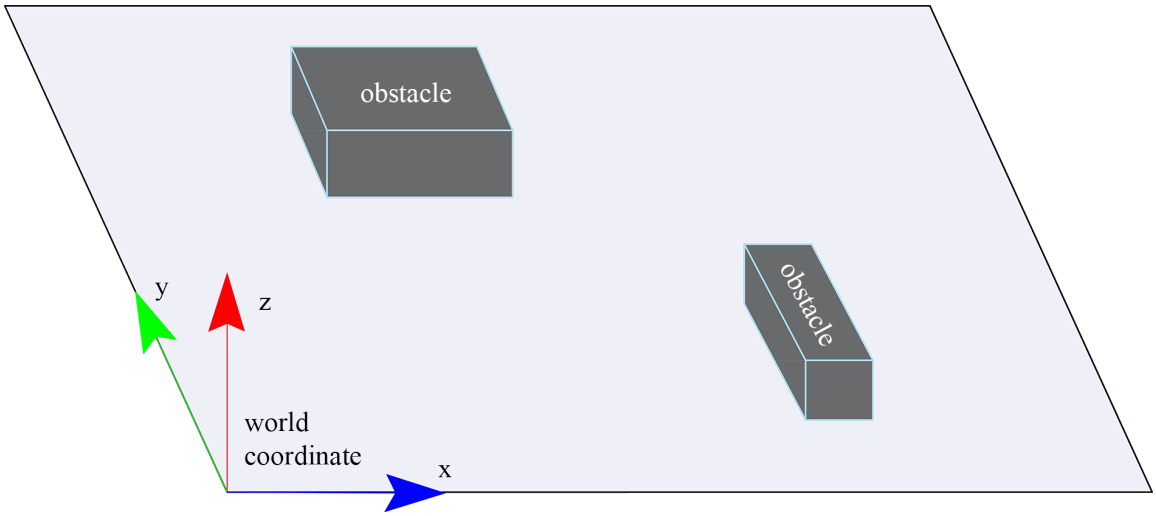


Figure 4.1 An example of the configuration space (world coordinate system $Oxyz$), composed of obstacles: the ground represents a grid for the path planning.

Assuming that the polyhedron has a state x and the subset of the workspace occupied by the polyhedron geometry is represented by $\mathcal{W}(x)$, the configuration space $\mathcal{O}_{obs}$ of the obstacle $\mathcal{S}_{obs}$ is a closed region in the $\mathcal{W}_{space}$ yielding the following

$$\mathcal{O}_{ct} = \{x \in \mathcal{W}_{space} : \mathcal{W}(x) \cap \mathcal{S}_{obs} \neq \emptyset\} \tag{4.1}$$

The idea of free space $\mathcal{W}_{free}$ is at the core of the path planning problem. Representation of $\mathcal{W}_{space}$ is at the discretion of the path planning algorithms. Path planning algorithms are generally categorized by the approach they take for representing the $\mathcal{W}_{space}$. The two main approaches are: explicitly computing the boundary of $\mathcal{O}_{obs}$ and sampling configurations from the $\mathcal{W}_{free}$. The former type of algorithm is called deterministic path planning algorithms. The focus of the deterministic approach is to explicitly compute the boundary of $\mathcal{W}_{free}$. Whether the computed boundary is exact or approximate, depends on the planner. The latter are called sampling-based

planning algorithms, they take random poses of the robot and see if they are not overlapping with the obstacles. In order words, the algorithms are operated by sampling the $\mathcal{W}_{free}$ and making a connected network from these sample poses. Collision detection is the main component of sampling-based approaches and is done to test whether a sample belongs to $\mathcal{W}_{free}$ or not.

When the polyhedron is capble of rotating while considering the aforementioned relationship between $\mathcal{O}_{ct}$ and the orientation of the polyon, $\mathcal{O}_{ct}$ become $3D$ subsets of $\mathcal{W}_{space}$. The $\mathcal{W}_{free}$ is formally defined as

$$\mathcal{W}_{free} = \mathcal{W}_{space} / \bigcup_{ct=1}^{n} \mathcal{O}_{ct} = \left\{ x \in \mathcal{W}_{space} : \mathcal{W}(x) \cap (\bigcup_{k=1}^{n} \mathcal{S}_{obs}) \neq \emptyset \right\} \tag{4.2}$$

4.2 BFS-Based Path Planning Algorithm

4.2.1 Introduction

The problem of finding the feasible path between two points has been a hot issue in many fields of science. Many previous studies have characterized the problem in linear programming terms [122] and dynamic programming terms [84] while pragmatic computer scientists have attempted to comprehend a simple but efficient computational method and have successful results [98, 123, 124]. The solution should not only

guarantee a minimum travelling distance without obstacle collision but also generate a smooth and feasible path from an initial configuration to a goal configuration. The initial environment for the path planning problem can be either static or dynamic. However, in the study of rolling polyhedra, only a static environment is considered to find a feasible path.

Based on the literature in Chapter 3, BFS is an algorithm used for traversing graphs or trees [125], which has the recursive property to search all the nodes of a tree. To compute the shortest paths from a single source node to every other node in a weighted graph, the Dijkstra algorithm [98] is the classical method for it. Most other algorithms for solving this problem are based on this algorithm but have improved data structures for implementation [126]. The paper [98] proposed using two variations of the classical shortest-path algorithms for link analysis. The evaluation studies assess both the effectiveness and efficiency of the proposed algorithms. The effectiveness issue concerns whether association paths found by the proposed algorithms are more useful for uncovering investigative leads than those found by a modified BFS algorithm. The modified BFS algorithm to a large extent simulated the manual approach of association search by crime investigators and was used as a benchmark technique for effectiveness comparison. The efficiency issue concerns which shortest-path algorithm is faster in what type of networks.

In summary, previous studies have proposed some techniques based on BFS for network construction in link analysis. However, little research has been done to address

the association search problem for rolling polyhedrons. Specifically, an effective and efficient link analysis from BFS is needed to find association paths between two or more source entities not directly related. Moreover, the paths found should reveal strong associations between entities so that important investigative leads can be uncovered. The BFS is proposed to use the shortest-path algorithms to achieve this goal. However, such an approach from previous studies cannot guarantee to find the strongest associations between entities and thus may not successfully generate investigative leads. In the next sections, the modified BFS algorithm is presented, which simulates the typical association search for rolling polyhedra.

4.2.2 BFS-based Path Planning Algorithm

The BFS algorithm is useful for analysing the nodes in a graph and constructing the shortest path of traversing through these. The technique can traverse through a graph in the smallest number of iterations. Furthermore, the architecture of the BFS algorithm is simple and robust while the result of the BFS algorithm holds a high level of accuracy in comparison to other algorithms. The BFS-based algorithm in this thesis is used to generate paths for the Platonic solids through edge-rolling from an initial pose to the desired pose on a plane. While at rest, a face of a Platonic solid is in contact with the plane, and the algorithm determines the edge, and subsequently the direction, of rolling.

Basic BFS algorithm

Breath-first search is similar to the depth-first search (DFS), except it attempts to make the search tree as broad as possible, as quickly as possible. This is accomplished by changing the open list into a queue, adding open nodes to the back of the queue, and expanding nodes off the front of the queue. This way, nodes are expanded in the same order they are discovered. BFS traversals typically require random memory accesses for checking if a neighbouring edge has previously been found. In opposition, determining if an edge is useful requires consecutive memory accesses, making it more efficient.

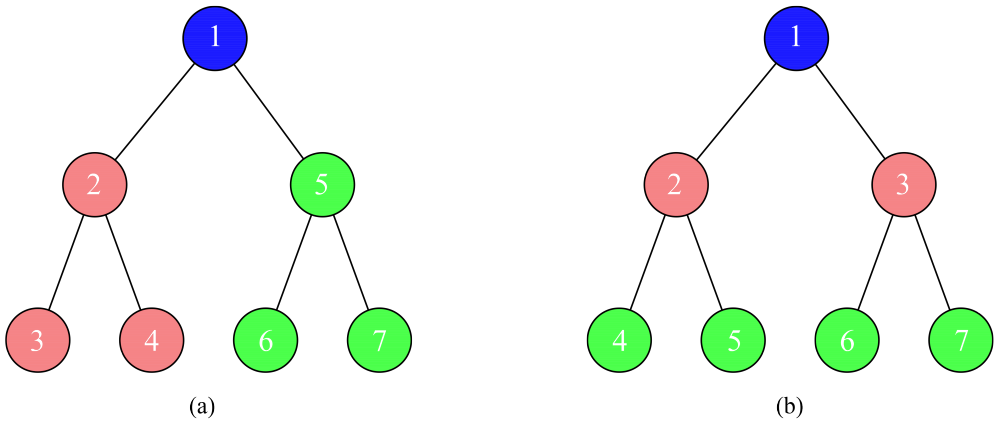


Figure 4.2 An overview of the graph search for Depth-First search (a) and the Breadth-First search.

Breadth-first search is complete in a finite graph. It is incomplete in a countably infinite graph; however, it will find a solution if one exists. The BFS algorithm is more methodical than the DFS algorithm and is usually more useful when the start configuration only includes a few nodes. Although the BFS search may be inefficient in large graphs and/or high-dimensional spaces, the path planning for the Platonic

solids fits on limited prescribed grids. The algorithm is useful for analysing the nodes in a graph and constructing the shortest path of traversing through these. BFS selects a single node (initial or source point) in a graph and then visits all the nodes adjacent to the selected node. The pseudo-code for the BFS algorithm is displayed in Algorithm 1.

In the beginning, the algorithm initialises the start and goal configuration with the input parameters with the number of edges and vertices. The Line 2 indicates that Set **O** is the *open* (active) frontier while Set **C** is the *closed* (inactive) set (Line 6). The *parent* pointers which creates spanning tree are used to enable path extraction (Line 3). The *while* loop executes the main BFS search on graphs with the sorting of the cost of optimality. But the sorting is trivial (FIFO) when the edges are uniform cost. It firstly generates all paths of length on edge then all paths of length two edges until reaching the goal (Return success in Algorithm 2).

Algorithm 1 Breadth-first search

```
1: Initialize: x_start, x_goal
2: O.insertLast(x_start)                          ▷ Insert at back
3: x_start.parent ← null
4: while O ≠ ∅ do                                 ▷ Terminate on failure
5:     x ← O.removeFirst()                        ▷ Remove at front
6:     C.insert(x)                                ▷ Move state behind frontier
7:     if (expandNodeBF(x)) 2 then
8:         return success
9:     end if
10: end while
11: return failure
```

Algorithm 2 expandNodeBF(x)

```
1: for each (u ∈ U(x)) do
2:     x_next ← f(x, u)
3:     if (x_next = x_g) then
4:         x_next.parent ← x;
5:         return success                                  ▷ Solution!
6:     else if (x_next ∉ O && x_next ∉ C) then
7:         x_next.parent ← x;                              ▷ Record backpointer
8:         O.insertLast(x_next)                            ▷ Insert at back
9:     end if
10: end for
11: return failure
```

Search Conditions

A graph search based on BFS has potential results in solving path planning problems to find the goal. The algorithm has some below conditions for searching.

- The BFS algorithm initiates at the root node and explores all adjacent graph edges, adding connected nodes to a first in, first out queue, as a data structure.
- Any node in the graph is marked as root and starts traversing the data from it.
- After discovering all connected nodes, the search moves to the first node in the queue and restarts the procedure. Then the search strategy moves to the next node in the queue after discovering all connected nodes.
- It is noted that the previously discovered nodes are not repeated. The iteration of the BFS algorithm will execute until traversing successfully all the vertices in the graph and then these vertices are marked as completed.

4.2.3 BFS Path Planning for Rolling Polyhedra

Node expansion is considered based on the unfolding polyhedron properties. Fig. 4.3, 4.4 shows examples of the basic unfolding step of a cube, a tetrahedron and a dodecahedron.

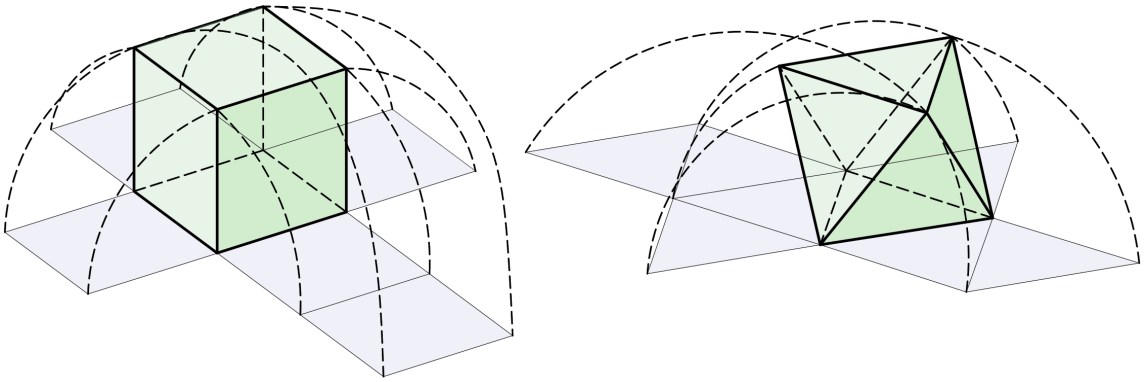


Figure 4.3 The way of unfolding cube and icosahedron onto a plane.

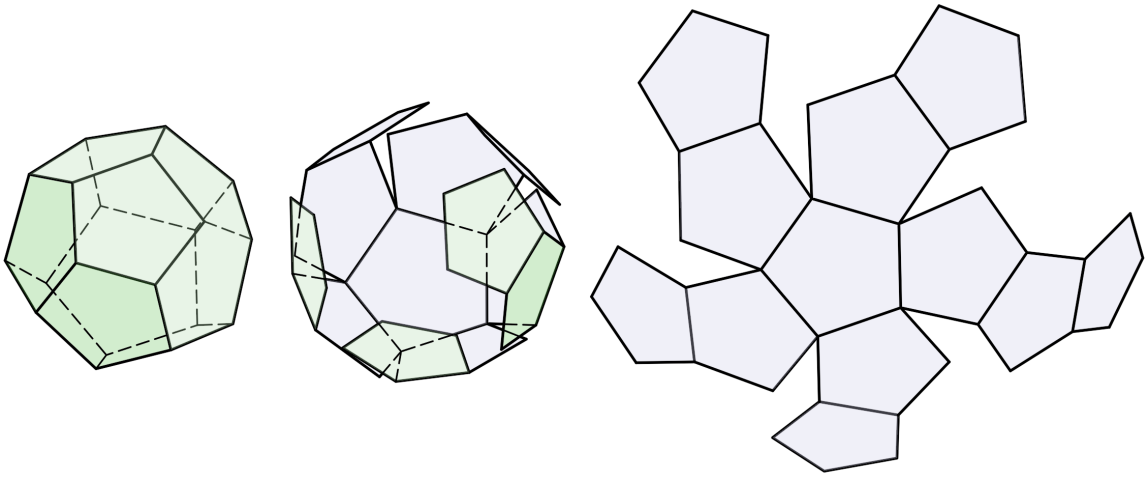


Figure 4.4 Dodecahedron unfolding method.

The potential rolling directions of each polyhedron will be checked from the unfolded polyhedra. The cube has four edge-contacts corresponding to the four rolling directions while the tetrahedron, octahedron and icosahedron have triangular surface contact with three edge contacts, which will consider for three rolling directions. The

dodecahedron with five edge contacts has five potential rolling directions. The general path-finding for the Platonic solids by using a BFS-based algorithm is mentioned in the tree expansion technique which is shown in the next section.

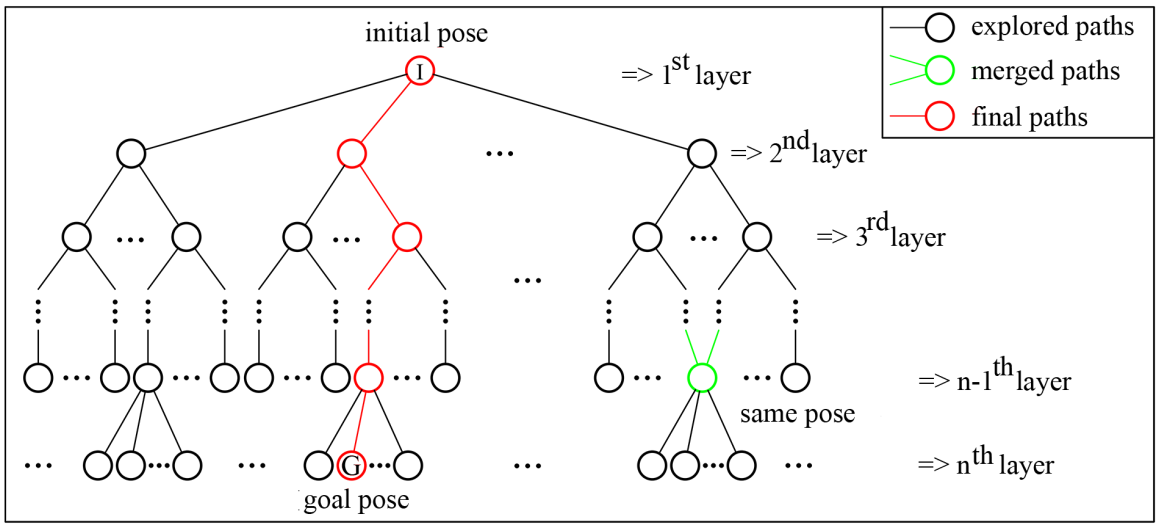


Figure 4.5 Tree exploration technique. Based on the BFS method, this algorithm starts from an initial pose represented by Node I. The branches represent the rolling directions for each iteration. If some nodes are of the same pose, they are merged to reduce the search space (the green node). The algorithm stops when the desired pose, represented by the Node G, is reached and the shortest path is generated (coloured in red).

Tree Expansion

The planning algorithm employs tree exploration (Fig 4.5), which is a variation of the BFS algorithm [125]. Using queues, this algorithm is faster than the $A^\star$ algorithm, which uses the priority queue [79], for the unweighted graph. Another advantage of the BFS algorithm is that it can find the shortest path where the environment is known. $A^\star$ can also implement to find the path but it requires a more general setting of weighted graphs. Thus, this section describes the tree expansion technique from the BFS-based algorithm as an efficient search algorithm to find the shortest path for rolling convex polyhedra on a $2D$ plane.

Nodes in the same layer representing the same pose are merged so that the algorithm only generates distinct paths. The checking condition of nodes' orientation between the current node and the previous nodes which have the same position is added in each search iteration. This step can reduce the latter time and space searching in the main algorithm. The first path from the initial pose I to reach the desired pose G is the shortest path because of the BFS algorithm. For the stopping criteria, it is applied for only the tetrahedron cases when the final tetrahedron's configuration reaches the target position but a different orientation.

Complexity in Time and Space Search

Based on tree traversal, BFS search has $O(m^{(n+1)})$ for time complexity and $O(m^n)$ for the space complexity, which is based on figuring out the size of a search tree and the number of nodes in a tree, where m is the maximum number of nodes in each search level and n is the number of layers. Depending on the type of the Platonic solids, $O(m^n)$ could be between $O(2^n)$ and $O(4^n)$ nodes in each layer. The first node I, which represents the initial pose, is the root of the tree, from which m nodes in the next layer are generated corresponding to m different directions of edge-rolling of each Platonic solid ($m = 3$ for tetrahedron, octahedron and icosahedron, $m = 4$ for cube, and $m = 5$ for dodecahedron) (Fig 4.5). From the newly generated nodes, each Platonic solid can only roll with $(m - 1)$ directions to avoid going back to the previous pose in the next layer.

4.3 RRT-Based Path Planning Algorithm

4.3.1 Introduction

A key advantage of the RRT technique is to handle high degrees of freedom problems for searching. RRT [108, 127] is a fast probability planning that can be extended to multidimensional spaces. The RRT is probabilistically complete [107] so that it guarantees the complete discovery and the exploration of the map. The RRT algorithm extends a tree structure from the starting point and follows an expansion direction, as determined through random selection points in the planning space. An opportunity constrained sub-path based on RRT can be generated from the starting position to the target [128]. Since the RRT algorithm is biased toward the undetected area [109], it can quickly detect the target on the map. RRT is also defined as a randomized data structure technique for solving planning problems, which can be applied for path planning problems under holonomic, nonholonomic and kinodynamic constraints [129]. RRT* improve the convergence rate but the planning time has significantly increased [130].

Rapidly-exploring random tree (RRT) algorithm is the most representative sampling-based path-planning algorithm. The RRT-based algorithm plans a path by gradually expanding a tree with a root node at the start point using random sampling. It is designed to handle non-holonomic constraints and high degrees of freedom. RRT path

planning is a potential algorithm to find a sub-optimal path for convex polyhedral through edge-rolling on a plane and this is the first time searching a rolling path for a truncated icosahedron.

4.3.2 Preliminaries

This section details the path planning based on the RRT method applied to find a path of rolling polyhedra from an initial pose (position and orientation) to a random goal (position and orientation) on a plane. It first introduces the notation and problem statement. Then it shows the basic RRT algorithm and its improvement used to find a path for a truncated icosahedron through edge-rolling contact on a plane.

Notations and Problem Statement

The polyhedron path planning problem is the task of finding the collision-free paths for rolling the polygon around static obstacles to achieve the goal configuration. A configuration of the polyhedron is a set of the positions of all the vertices of the polyhedron relative to a world coordinate system. Let $\mathcal{W}$ denotes the configuration space of the polyhedron in which $\mathcal{O}_{obs}$ is the obstacle region in the environment, such that $\mathcal{W} \setminus \mathcal{O}_{obs}$ is an open set. Workspace $\mathcal{W}$ is a subset of $\mathbb{R}^2$ or $\mathbb{R}^3$, and denotes the obstacle-free region is $\mathcal{W}_{free}$. In this study, although the rolling path is considered on a plane, the orientation of the polyhedra changes in $3D$ space. We then consider

$\mathcal{W} \subseteq \mathbb{R}^3$. The tree is denoted by $\mathcal{T}$ and the node of the tree - including position and orientation in the environment, is defined by $q_i \in \mathcal{W}_{free}$. The initial configuration q_{start} is an element of $\mathcal{W}_{freedom}$ and the a random goal q_{goal} is an open subet of $\mathcal{W}_{free}$. In the path planning algorithm the set of the planned configurations is defined by $(q_0, q_1, ..., q_n)$ in which n is the limited path node running from q_0. The straight continuous path between two nodes $q_i, q_{i+1} \subseteq \mathbb{R}^3$ is represented by DISTANCE(q_i, q_{i+1}). If we denote $\overline{q_i q_{i+1}}$ as the connection node between the two configurations (q_i and q_{i+1}), the results of $\overline{q_i q_{i+1}} \in \mathcal{W}_{free}$ define the successful extension and $\overline{q_i q_{i+1}} \notin \mathcal{W}_{free}$ define the failure extension.

The algorithm shows whether a path is feasible or fails to meet the goal. If a path exists, the tree $\mathcal{T} = (V, E)$ on $\mathcal{W}$ represents a set of vertices V sampled from $\mathcal{W}_{free}$ and the edges E connect these vertices, such that V is a finite subset of $\mathcal{W}$, and E is a subset of $V \times V$. A direct graph on $\mathcal{T}$ is a sequence $(v_0, v_1, ..., v_n)$ of vertices such that $v_i, v_{i+1} \in E$ for all $1 \leq i \leq n-1$. Given a vertex $v \in V$, the sets $\{u \in V | (u, v) \in E\}$ and $\{u \in V | (v, u) \in E\}$ are said to be its incoming neighbors and outgoing neighbors, respectively. A (directed) tree is a directed graph, in which each vertex but one has a unique incoming neighbor; the vertex with no incoming neighbor is called the root vertex. Vertices of a tree are often also called nodes.

The core of the polyhedron rolling path planning problem is to find the optimal path to achieving the target configuration. The discretization of the path planning algorithm is represented by the sampling configuration from the $\mathcal{W}_{free}$. First, the path from q_{start} to q_{goal} is found with precise position and orientation. However, due to symmetrical properties and tiling polyhedrons, path planning does not always find the solution. An algorithm is developed to find a closed path in which the final node q_n gets closest to the goal q_{goal} with $q_n \in \mathcal{G}_{goal}$ ($q_{goal} \in \mathcal{G}_{goal}$ the goal region). The rolling of a polyhedron by edge-contact will be considered as a nonholonimic path planning, then we need a threshold $\parallel q_{goal} - q_n \parallel \leq \varepsilon$ for a small $\varepsilon > 0$. Considering a minimum length path for rolling polyhedra to achieve the goal, we define the minimum expected cost (denoted by $\mathcal{C}_i = f_i + h_i$) to plan a path from q_{start} to q_n close to q_{goal}. The optimal path has the lowest average cost compared to other possible paths on the tree graph. The Euclidian length of the path from q_i to q_{i+1} is computed. The length may vary after the rolling motion due to different polyhedra shapes.

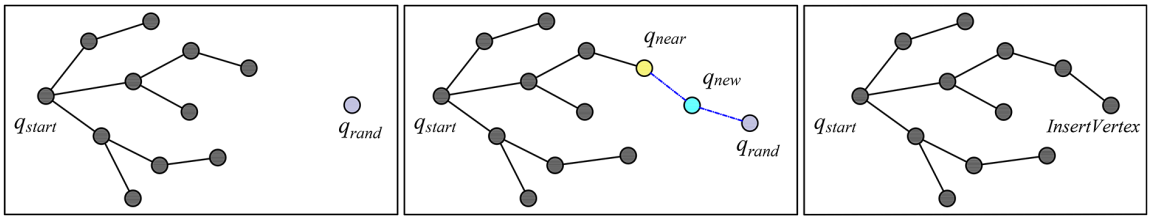


Figure 4.6 Basic rapidly exploring random tree algorithm without obstacle in one iteration. Left: randomly generating a configuration q_{rand}. Center: the nearest configuration in the existing $\mathcal{T}$ tree q_{near} is selected for expansion. A new node q_{new} is created by running the shortest distance to new node $StraightLine(q_i, q_{i+1})$. Right: The tree is updated with the new configuration.

RRT Path Planning

The essential RRT algorithm is formed by sampling random points in state space and extending the RRT toward them in high-dimensional spaces. The key idea is to bias exploring the search tree away from the root, q_{start} to a goal region $\mathcal{G}_{goal} \subset \mathcal{W}$ or a goal state $q_{goal} \subset \mathcal{G}_{goal}$. The following steps then are repeated until a feasible path is generated, or the search planning approaches a limit condition such as a maximum number of iterations or memory or time limits. A configuration $q_{rand} \in \mathcal{W}$ is randomly sampled in the configuration space. The nearest neighbor q_{near} in the exploring tree is selected by using shortest distance $StraightLine(q_i, q_{i+1})$ checking condition. Then an edge of the connection between q_{near} to q_{new} is developed. The q_{new} configuration is inserted onto the tree $\mathcal{T}$ as a child of the q_{near} if $q_{new} \in \mathcal{W}_{free}$ (Fig. 4.6).

Algorithm 3 Basic Rapidly-exploring random tree (RRT) Expansion

Input: Initial configuration: $V \leftarrow q_{start}$, $E \leftarrow \emptyset$, iterations N, incremental distance δ_q
Output: RRT graph $\mathcal{T} = (V, E)$

while $i < N$ **do** ▷ The iteration is limitted by N
  $q_{rand} \leftarrow RandomState()$
  $q_{near} \leftarrow NearVertex(q_{rand}, \mathcal{T} = (V, E))$
  $q_{new} \leftarrow NewState(q_{near}, q_{rand}, \delta_q)$ ▷ Update the new state
  if q_{new} can connect to q_{near} **then**
    $V \leftarrow InsertVertex(q_{new})$
    $E \leftarrow InsertEdge(q_{near}, q_{new})$
    $T \leftarrow (V, E)$ ▷ Store the vertices and edges into the tree
  end if
end while
return $\mathcal{T}$

The RRT algorithm is the primary of single-query planner which is one the category of sampling-based motion planning algorithms in the literature. The basic RRT algorithm is outlined in Algorithm 3. The algorithm starts with the tree $\mathcal{T}$ which includes the initial configuration as an intial vertex q_{start} and no edges; then, they incrementally grow the tree on $\mathcal{W}_{free}$ by sampling a state $q_{rand} \in \mathcal{W}_{free}$ at random and extending the tree towards q_{rand}. The single iteration of the incremental sampling-based algorithm leads to a rapid exploration (Steps 1-10 of Algorithm 3).

Algorithm 4 RRT Path Planning with Obstacle Avoidance

Input: $q_{start}, q_{goal}, \mathcal{O}_{obs}, N$
Output: RRT graph $\mathcal{T} = (V, E)$
1: *Initialisation* :$\mathcal{T} \leftarrow (q_{start}, q_{goal}, \mathcal{O}_{obs})$
2: **for** $i = 1...N$ **do**
3: $q_{rand} \leftarrow RandomState()$
4: $q_{near} \leftarrow NearVertex(q_{rand}, \mathcal{T})$
5: $q_{new} \leftarrow NewState(q_{near}, q_{rand}, \delta_q)$ ▷ Update a new state
6: **if** ObstaclesFree(q_{near}, q_{new}) and MinExpandCost() **then** ▷ Check the obstacle free and lowest cost for node expansion
7: $V \leftarrow V \cup \{q_{new}\}$
8: $E \leftarrow E \cup \{(q_{nearest}, q_{new})\}$
9: $\mathcal{T} \leftarrow (V, E)$ ▷ Store the nodes into the tree
10: **end if**
11: **end for**
12: **return** $\mathcal{T}$

The principle procedures of the extended RRT path planning algorithm including the obstacle avoidance step is shown in Algorithm 4. For a given initial state q_{start} and a distance metric on the state space δ, the initial vertex of $\mathcal{T}$ is $q_{start} \in \mathcal{W}_{free}$. In each iteration $i < N$ (N is the limited searching iteration), a random state - q_{rand} - is selected from $\mathcal{W}$. The $NearVertex(q_{rand}, \mathcal{T})$ from the step 4 will find and store the closest vertex to q_{rand} in terms of δ. Step 5 stores the new state q_{new} which is added as a vertex to $\mathcal{T}$. Then an edge from q_{near} to q_{new} is also added through

the function $InsertEdge(q_{near}, q_{new})$. This step also integrates collision detections in the presence of convex obstacles (Step 6). A null configuration is returned and the extension fails when the new portion of the path leads to a collision independently of the associated costs. This extension process ensures the bias toward unexplored free regions of space. The goal of this step is also to filter irrelevant configurations regarding the search for low-cost paths before inserting q_{new} and $q_{nearest}$ into the tree $\mathcal{T}$.

The path planning algorithms based on RRT for rolling contact have the ability to explore the whole state space $\mathcal{W}_{space}$. The step is achieved by computing the boundary of $\mathcal{O}_{obs}$ and random sampling q_{rand} from $\mathcal{W}_{free}$. A graph in Fig. 4.7 represents one iteration of the tree expansion with a fixed obstacle. The connection between $q_{nearest}$ to q_{new} is found after checking collision between the obstacle and the q_{near}. The updated edge will be stored in E by using the $InsertEdge$ function.

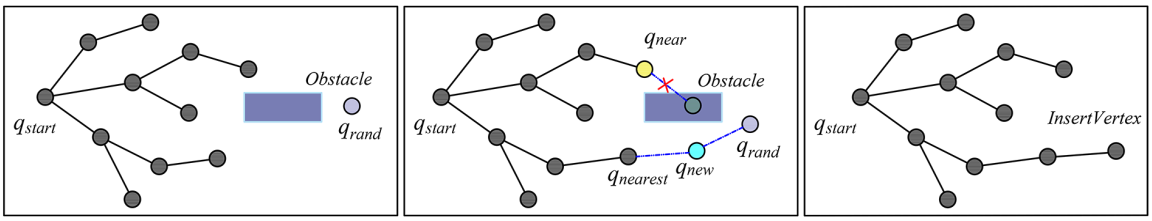


Figure 4.7 Tree expansion from RRT with obstacle avoidance in one iteration. Left: randomly generating a configuration q_{rand}. Center: the nearest configuration $q_{nearest}$ in the existing $\mathcal{T}$ tree is selected for expansion after the q_{near} is found in $\mathcal{O}_{obs}$. A new node q_{new} is created by running the shortest distance to new node $StraightLine(q_{nearest}, q_{new})$. Right: The tree is updated with the new configuration.

4.3.3 RRT Path Planning for Convex Polyhedra

The path planning algorithm for the rolling polyhedra task is developed from the RRT algorithm, which is considered a nonholonomic motion planning system (Algorithm 5). The initial environment includes initialized polyhedron model $\mathcal{M}_{Poly}$ (Step 1) such as the polygon type, its centre, vertices, dimension of each edge and the obstacles. The algorithm interleaves path planning with tree expansion and path rewiring to achieve the random goal configuration with obstacle avoidance.

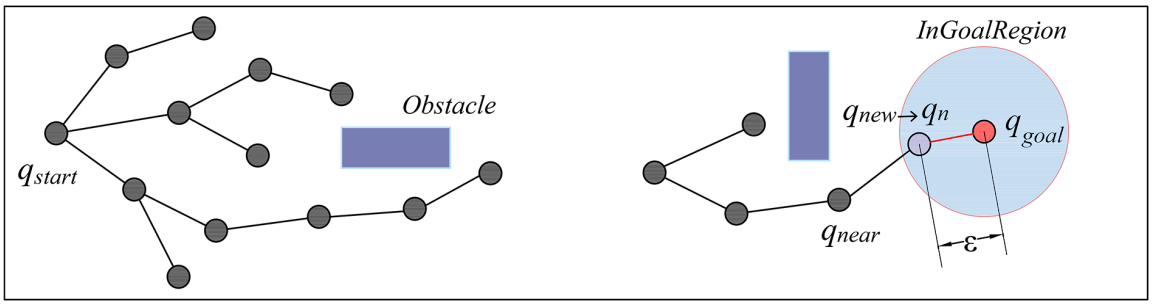


Figure 4.8 Improved RRT path planning to achieve a goal state q_{goal} at a closest q_n state.

The main part of the path planning algorithm is to achieve the closest tartget orientation through the edge-rolling contact of the truncated icosahedron. The tree expansion is done by selecting a n^{th} state ($q_n \in \mathcal{G}_{goal}$) which is closest to the q_{goal} due to the geometrical properties of different polyhedra (Fig. 4.8). The last configuration can not always reach the exact position and orientation of the goal configuration. Then a threshold ε is defined to the goal configuration so that the final state which reachs the InGoalRegion() returns the successful path (Step 15). The state q_n should meet at least two conditions including a correct orientation as the target orientation q_{goal} and $||q_n - q_{goal}|| \leq \varepsilon$ to return the $True$ value in the InGoalRe-

Algorithm 5 Path Planning Based on RRT for Rolling Polyhedron with Obstacles Avoidance

Initialisation:

1: $\mathcal{M}_{Poly}$
2: $V \leftarrow \{q_{start}\}$ and $E \leftarrow \emptyset$
3: $\mathcal{Q}_{soln}$ and $\mathcal{O}_{obs}$
4: $\mathcal{T} \leftarrow \{V, E, \mathcal{M}_{Poly}\}$ ▷ Initialise the polyhedron to the tree $\mathcal{T}$
5: $N \leftarrow$ iteration number
6: **while** $i < N$ **do**
7: $\mathcal{M}_{Poly} \leftarrow \{V, E, Center, Orientation\}$
8: $\mathcal{C}_i \leftarrow min_{q_{soln} \in \mathcal{Q}_{soln}}\{\textsc{Cost}(q_{soln}, q_{goal}, \mathcal{M}_{Poly})\}$ ▷ Calculate the heuristic cost
9: $q_{rand} \leftarrow \textsc{Sampling}(q_{start}, q_{goal}, \mathcal{C}_i)$
10: $q_{nearest} \leftarrow \textsc{Nearest}(q_{rand}, \mathcal{T})$
11: $q_{new} \leftarrow \textsc{Steer}(q_{nearest}, q_{rand}, \delta_q)$
12: $\mathcal{M}_{Poly} \leftarrow \textsc{PolyRoll}(\mathcal{M}_{Poly}, \alpha_{Poly})$ ▷ Take rolling action for the polyhedron
13: **if** $\textsc{CollisionFree}(q_{nearest}, q_{new}, \mathcal{O}_{obs})$ **then**
14: `TreeExpansion` using Algorithm 6
15: **if** $\textsc{InGoalRegion}(q_n, q_{goal}, \varepsilon)$ **then**
16: $\mathcal{Q}_{soln} \leftarrow \mathcal{Q}_{soln} \cup \{q_n\}$
17: $\textsc{PathRetrieve}(\mathcal{T}, \mathcal{Q}_{soln})$ using Algorithm 7
18: **end if**
19: **end if**
20: **end while**
21: **return** $\mathcal{T}, \mathcal{M}_{Poly}$

GION() function. Once the initial path is reached in this function, the cost COST(()) function will return the cost of the path, which is defined by Euclidean distance $\Delta d = \sqrt{(x_{q_j} - x_{q_i})^2 + (y_{q_j} - y_{q_i})^2}$.

As the solutions are found (Step 15 - Algorithm 5), the improved algorithm adds the results to a list of feasible solutions (Step 16 - Algorithm 5). These lists can be reduced by using the minimum at Step 8 - Algorithm 5. Subfunctions from the algorithm are described below.

Algorithm 6 Tree expansion

Input: $q_{nearest}, q_{new}, \mathcal{O}_{obs}$
Output: $q_n, \mathcal{T}, \mathcal{M}_{Poly}$

$V \leftarrow V \cup \{q_{new}\}$
$\mathcal{Q}_{near} \leftarrow \textsc{Near}(q_{new}, \mathcal{T}, \mathcal{M}_{Poly})$
$q_{min} \leftarrow \textsc{UpdateParent}(q_{nearest}, q_{near}, q_{new})$
$\mathcal{C}_{min} \leftarrow \textsc{Cost}(q_{min}) + \mathcal{C} \cdot \textsc{Distance}(q_{nearest}, q_{new}) + \mathcal{C} \cdot \textsc{PolyRoll}(\mathcal{M}_{Poly}, \alpha_{Poly})$
for all $q_{near} \in \mathcal{Q}_{near}$ **do**
    $\mathcal{C}_{new} \leftarrow \textsc{Cost}(q_{new}) + \mathcal{C} \cdot \textsc{Distance}(q_{near}, q_{new}) + \mathcal{C} \cdot \textsc{PolyRoll}(\mathcal{M}_{Poly}, \alpha_{Poly})$
    if $\mathcal{C}_{new} < \mathcal{C}_{min}$ **then**
        if $\textsc{CollisionFree}(q_{near}, q_{new}, \mathcal{O}_{obs})$ **then** ▷ Check the obstacle avoidance
            $q_{min} \leftarrow q_{near}$
            $\mathcal{C}_{min} \leftarrow \mathcal{C}_{new}$
        end if
    end if
end for
$E \leftarrow E \cup \{(q_{new}, q_{min})\}$
$\mathcal{T} \leftarrow (E, V)$
for all $q_{near} \in \mathcal{Q}_{near}$ **do**
    $\mathcal{C}_{near} \leftarrow \textsc{Cost}(q_{near})$
    $\mathcal{C}_{new} \leftarrow \textsc{Cost}(q_{new}) + \mathcal{C} \cdot \textsc{Distance}(q_{near}, q_{new}) + \mathcal{C} \cdot \textsc{PolyRoll}(\mathcal{M}_{Poly}, \alpha_{Poly})$ ▷ Calculate the new heuristic cost
    if $\mathcal{C}_{new} < \mathcal{C}_{near}$ **then**
        if $\textsc{CollisionFree}(q_{near}, q_{new}, \mathcal{O}_{obs})$ **then**
            $q_{parent} \leftarrow \textsc{UpdateParent}(q_{near})$
            $q_n \leftarrow q_{new}$
            $\mathcal{T} \leftarrow \textsc{InsertVertex}(q_{parent}, q_n, \mathcal{T})$
            $\mathcal{T} \leftarrow \textsc{Replanning}(\mathcal{T}, q_{near}, q_{min}, q_{new})$ ▷ Call back the plan function to update the new nodes
            Nodes_results$\leftarrow \mathcal{T}$ ▷ Save the nodes
        end if
    end if
end for

- Sampling (Step 9 - Algorithm 5): The function Sampling given two poses, $q_{start}, q_{goal} \in \mathcal{W}_{free}$ and a best heuristic value, $\mathcal{C}_i \in \mathbb{R}$, the function Sampling $(q_{start}, q_{goal}, \mathcal{C}_i)$ returns independent and identically distributed samples from state space $\mathcal{W}_{free}$. The heuristic computes the distance between q_{near} and q_{new}

in the obstacle free space. Based on the dimensions of the plane and obstacles, the convex sampling emvironment is found so that the random sampling nodes are inside it.

- Nearest Neighbor (Step 10 - Algorithm 5): Given a graph $G = (V; E)$, and a point $q_n \in \mathcal{W}_{free}$, the function NEAREST$(q_{rand}, \mathcal{T})$ will return the closest point $q_{nearest}$ to q_n in terms of a given DISTANCE function. This study uses Euclidean distance to find distance between any two points in $\mathcal{W}$.

- Near Vertices (Step 2 - Algorithm 6): The function NEAR$(q_n, \mathcal{T}, \mathcal{M}_{Poly})$, given by a graph $G = (V; E)$, a point $q_n \in \mathcal{W}_{free}$, returns a set of nearby neighbor nodes. This function differs from the NEAREST function which finds only a closest vertex. The result from NEAR function should lie in a sphere volume of radius $r_{sphere} = \mu(\frac{\log n}{n})$.

- Steering (Step 11 - Algorithm 5): The function STEER drives the system from $q_i = q_{rand}$ to $q_{i+1} = q_{nearest}$ along the path. The two given points returns a new point $q_{new} \in \mathbb{R}^3$ such that q_{new} is closer to q_{x+1} than to q_i. Another DISTANCE function, contains the cost $\mathcal{C}$ with Euclidean distance, is used to returns the cost of the path between these two states while satisfying COLLISIONFREE condition. Then, differential relationship $||q_{new} - q_{i+1}|| \leq \varepsilon$ is maintained.

- CollisionFree (Step 13 - Algorithm 5 and Step 20 - Algorithm 6): The regions of all obstacles $\mathcal{O}_{obs}$ are considered first in this step. From the tree expansion, with any two closed points $q_i, q_{i+1} \in \mathcal{W}$, running the collision checking of the

Boolean function CollisionFree($q_i, q_{i+1}, \mathcal{O}_{Obs}$) will return $True$ if the line segment between q_i and q_{i+1} lies in $\mathcal{W}_{free}$, and $False$ with intersection results.

- Insert Vertex (Step 23 - Algorithm 6): The function InsertVertex ($q_{parent}, q_n, \mathcal{T}$) adds the node q_n after having updated parent_nodes q_{parent} to V. This step connects a new node to parent nodes and adds the edge to E. The cost is found based on the minimum distance between q_{near} and q_{new} which is epxlored with tree expansion process and the cost of rolling polyhedron with checking the orientation.

- PolyRoll (Step 5,6,18 - Algorithm 6): This function PolyRoll requires input values of the current configuration of the polyhedron and the rotation angle α_{Poly} to achieve the next configuration.

- InGoalRegion (Step 15 - Algorithm 5): Given a pose, $q_n \in \mathcal{W}_{free}$, the function InGoalRegion(q_n) returns a $True$ value if and only if the state q_n lies in the goal region $\mathcal{G}_{goal}$, and has the same orientation as the target; otherwise it returns $False$. The condition to check for this step uses ε in which $||q_n - q_{goal}|| \leq \varepsilon$.

- Cost(q_i, q_j): This function evaluates the cost of transition from state q_i to q_j. The function is designed to generate the paths that suitable for edg-rolling contact through each PolyRoll action (taking rolling action for the polyhedron in α_{Poly}.

Shortest Path and Complexity

More formally, a given graph $\mathcal{T} = (V, E)$ with a length function $l : E \to R > 0$, and a vertex $v \in V$, the shortest path tree for $\mathcal{T}$, l, and v can be found in time $O(|V|log(|V|) + |E|)$. Let $\mathcal{T} = (V, E)$ be a graph with $V \in X$ and let $x \in X$. The number of simple operations executed by the $Nearest(\mathcal{T}, x)$ procedure $O(log|V|)$ in fixed dimensions. In Algorithm 5, complexity of the $CollisionFree$ procedure (Step 13) in terms of the number of obstacles in the environment is analyzed, which is a widely studied problem in the literature ([131]).

Tree Expansion

The core of path planning based on RRT algorithm for rolling polyhedron represented in Algorithm 5 is the `TreeExpansion` (Algorithm 6). The function CollisionFree is always checked in the loop to confirm free space $\mathcal{W}_{free}$ for expanding the tree nodes. The tree stores the new state along with the current polygon configuration that will update the parent nodes. The tree consists of multiples nodes which can be represented as $\mathcal{T} = \{q_0, ..., q_n\}$ where state q_0 is same as the start configuration q_{start} while the state q_n is exact state of the goal configuration q_n or the closest target configuration. Whenever a new state q_{new} is found, its configuration are compared to the $q_{nearest}$ before the parent node stores this new node. Then the path is retrieved from the goal to the start node when the $\mathcal{Q}_{soln}$ is found. The tree expansion process

is repeated until finding the q_n which meets the conditions of the InGoalRegion function.

The RRT path planning based algorithm in this study is used for rolling polyhedron as the nonholonomic system. This means that the next rolling positions are limited due the polygon contact-based shapes. For example, there are six potential states in the next rolling for the truncated icosahedron with the current hexagon based and only five states for the pentagon face contact. Before adding the new node to the tree, the minimum cost value will be calculated based on the cost of checking minimum length from the q_{near} node to the goal, the cost of the new node q_{new} to the current node and the cost of the polyhedron orientation. Reducing the convergence time of the tree expansion process is achieved by choosing a theshold ε small enough to attain the goal configuration with acceptable precision. In this study, the random state is selected to best speed up convergence. A goal region $\mathcal{G}_{goal}$ is defined as a sphere which has a volume greater than or equal to the threshold ε as shown in Fig. 4.8.

Algorithm 7 Reconstruction Path

```
1: function PathRetrieve(T, q_n)
2:     T ← ∅
3:     T.resize(q_n)                          ▷ Initialise the size of the tree
4:     while Nodes_results[q_n] ≠ NULL do
5:         q_n ←Nodes_results[q_n]
6:         T.push_front(q_n)      ▷ Insert the nodes in the list container at the front
7:     end while
8:     return T
9: end function
```

After finding the closest configuration $q_n \in \mathcal{G}_{goal}$, the function PathRetrieve from Algorithm 7 will reconstruct the rolling path from q_n to q_{start} to implement the rolling truncated icosahedron on the plan. The Nodes_results is a temporary matrix which saves the planned nodes whenever the path planning method finds the best node. Then they will be stored in the tree $\mathcal{T}$ as the final path for rolling. The path planning through edge-rolling of the truncated icosahedron is detailed in the Section 6.4.

4.4 Conclusion

The fisrst part of this chapter analyses the BFS algorithm which works on a similar principle of a graph traversal. The iteration of the algorithm is a unique process that requires the algorithm to visit, check, and/or update every single unvisited node in a tree-like structure. The BFS Algorithm has a wide range of real-world applications due to well perfomance in the small searching space. The algorithm traverses the graph in the smallest number of iterations and the shortest possible time to find the paths. Because of the advantages of the BFS algorithm including finding the shortest path or the solution with minimal steps if there are more than one solution, the BFS is the potential algorithm to solve the path planning for the Platonic solids through edge-rolling on prescribed grids. Based on the literature review, it is the first time to use BFS algorithm to search the rolling-path for Platonic solids on a prescribed grids, which is presented in the next chapter.

In another hand, the RRT-based algorithm is represented in the second part of the chapter. The algorithm is a kind of probabilistically complete exploration algorithm based on the tree structure. It has been widely used in the path planning problem since it guarantees the complete discovery and the exploration of environment maps through objects. In the present study, the RRT algorithm is extended to propose a path finding strategy for convex polyhedra to actively explore and find optimal path. From the RRT-based algorithm, the cost function consists the unknown region and the passed unknown region. The unknown region is explored for a given goal pose, while the passed unknown region is the area, where the polyhedron moves towards the target frontier point. This is a potential discrete path-finding algorithm which can explore and find paths for rolling-polyhedra through edge-rolling on a plane, which is presented in Chapter 6.

Chapter 5: BFS Path Planning for Platonic Solids through Edge-rolling on prediscribed grids [1]

The five Platonic solids—tetrahedron, cube, octahedron, dodecahedron, and icosahedron—have found many applications in mathematics, science, and art. Path planning for the Platonic solids had been suggested, but not validated, except for solving the rolling-cube puzzles for a cubic dice. We developed a path-planning algorithm based on the breadth-first-search algorithm that generates a shortest path for each Platonic solid to reach a desired pose, including position and orientation, from an initial one on prescribed grids by edge-rolling. While it is straightforward to generate triangular and square grids, various methods exist for regular-pentagon tiling. We chose the Penrose tiling because it has five-fold symmetry. We discovered that a tetrahedron could achieve only one orientation for a particular position.

[1]This chapter is reproduced from the journal paper: Lam NT, Howard I, Cui L (2021) Path planning for the Platonic solids on prescribed grids by edge-rolling. PLoS ONE 16(6): e0252613. https://doi.org/10.1371/journal.pone.0252613

5.1 Introduction

Planning techniques are categorized into different aspects. The basic idea of discrete path planning in the most cases is that state-space models will be used to demonstrate the distinct situation in which the task of a planning algorithm solves the sequence actions transforming from a initial state to other states [1]. For example, Thomas [73] applied Delaunay triangulations to discretize the environment, and cubic spline representations are proposed to meet robot kinematic constraints. Considering the continuous curvature on smooth curves has been integrated within the probabilistic approaches in order to compute the piecewise smooth paths for a car-like vehicle as a four-dimensional system [74]. Whereas, dealing with nonholonomic constraints, a sampling-based road map technique was proposed in [106]. Based on decomposing space into cells [132], a potential field without local minima was assigned with polygonal partitions of planar environments to solve the Laplace's equation problems in each cell exist.

Literature on path planning for polyhedra by edge-rolling is scarce. An attempt was made to plan a path for an octahedron edge-rolling on a plane from an initial pose (position and orientation) to a desired pose, which, unfortunately, failed due to errors propagating from the algorithm [38]. In graspless manipulation, two movable parallel plates working as a robotic end-effector rolled a cubic dice by edges [18], but this work did not discuss how to generate the desired path. The rolling-cube puzzles, which

focus on how to roll a cubic dice on a board consisting of labeled and white cells, were solved by detecting a Hamiltonian path in grid graphs as an NP-hard problem [69]. In this work, we propose a path-planning algorithm using tree exploration for each of the five Platonic solids starting from an initial pose to a desired pose by edge-rolling on different prescribed grids. We believe this is the first work that solves this problem.

This chapter is organized as follows. Firstly, the related works for the Platonics solids and the different patterns of grids are briefly reviewed. Secondly, the path planning algorithm is described. Then, the results for the proposed algorithm is presented. Finally, the chapter is concluded.

5.2 Platonic Solids

The Platonic solids is detailed in the Chapter 3 at the Section 3.2.2. This section describes different patterns of grids used to implement a path planning algorithm. Each of the Platonic solids can be unfolded into non-overlapping edge-joining polygons (Fig 5.1). The cube consists of 6 square polygons; the tetrahedron consists of 4 equilateral triangular polygons; the dodecahedron consists of 8 equilateral triangular polygons; the icosahedron consists of 20 equilateral triangular polygons; the dodecahedron consists of 20 hexagons and 12 pentagons.

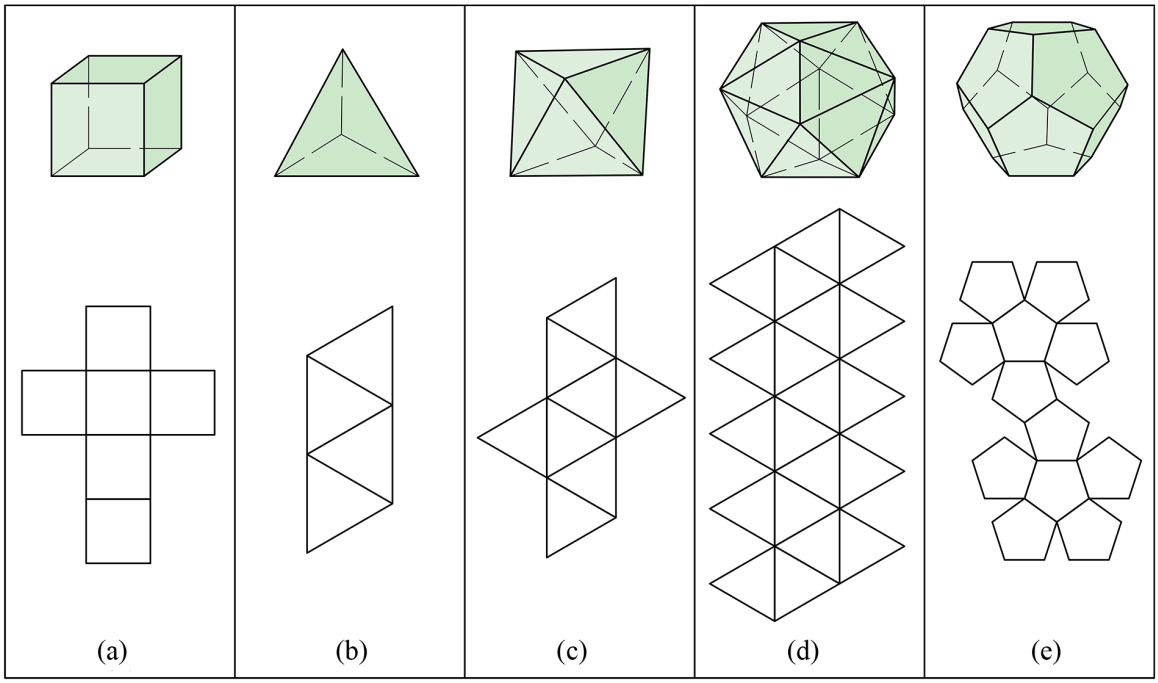


Figure 5.1 Five models of the Platonic solids and their unfolding geometry. (a) Cube. (b) Tetrahedron. (c) Octahedron. (d) Icosahedron. (e) Dodecahedron.

5.2.1 Discretized Grids

A plane can be discretized into a square, triangular, or pentagon grid, depending on the face of a Platonic solid in contact with it (Fig 5.2). At any instant, a cube has 4 edges in contact with the plane, which indicates 4 possible directions of edge-rolling on the square grid; a tetrahedron, octahedron, or icosahedron has 3 edges in contact with the plane, which indicates 3 possible directions of edge-rolling on the triangular grid; a dodecahedron has 5 edges in contact with the plane, which indicates 5 possible directions of edge-rolling on the pentagon grid.

There are many options for discretizing a plane into a pentagon grid. Regular pentagons tiling a plane will leave symmetric gaps without overlap (Fig 5.3). A variety

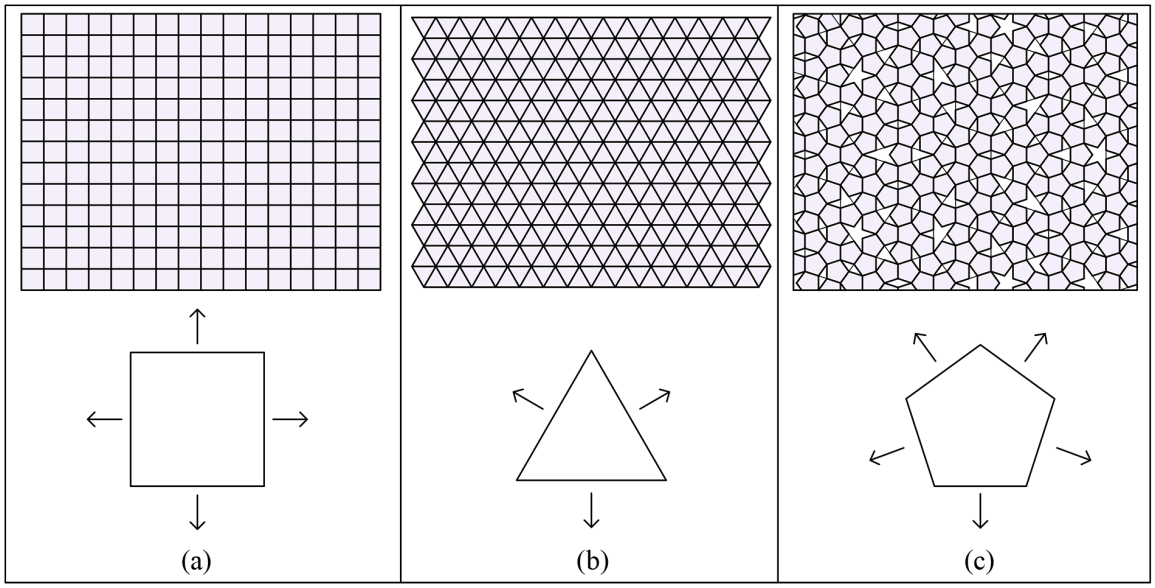


Figure 5.2 Direction of grids. Three different patterns of grid of the plane for the Platonic solids. (a) A square grid for the cube with 4 edge-rolling directions. (b) A triangular grid for the tetrahedron, octahedron, and icosahedron with 3 edge-rolling directions. (c) A pentagon grid using Penrose tiling for the dodecahedron with 5 edge-rolling directions.

of patterns exist, such as those developed by Dürer (Fig 5.3(a) and 5.3(b)) [136], Caris (Fig 5.3(c), 5.3(d) and 5.3(e)) [31], and Penrose (Fig 5.3(f)) [137]. The Dürer and Caris tiling uses multiple twins of regular pentagons to tile a plane, in which rhombi remain between pentagons in various positions. The Penrose tiling attaches five regular pentagons onto the initial one along its edges to form a new larger pentagon, which generates gaps in the shapes of rhombi, pentacles, and half-pentacles. These gaps are partially filled by inserting pentagons following substitution rules (Fig 5.4) [138]. To facilitate path planning, we chose Penrose tiling because it has five-fold symmetry, which others lack.

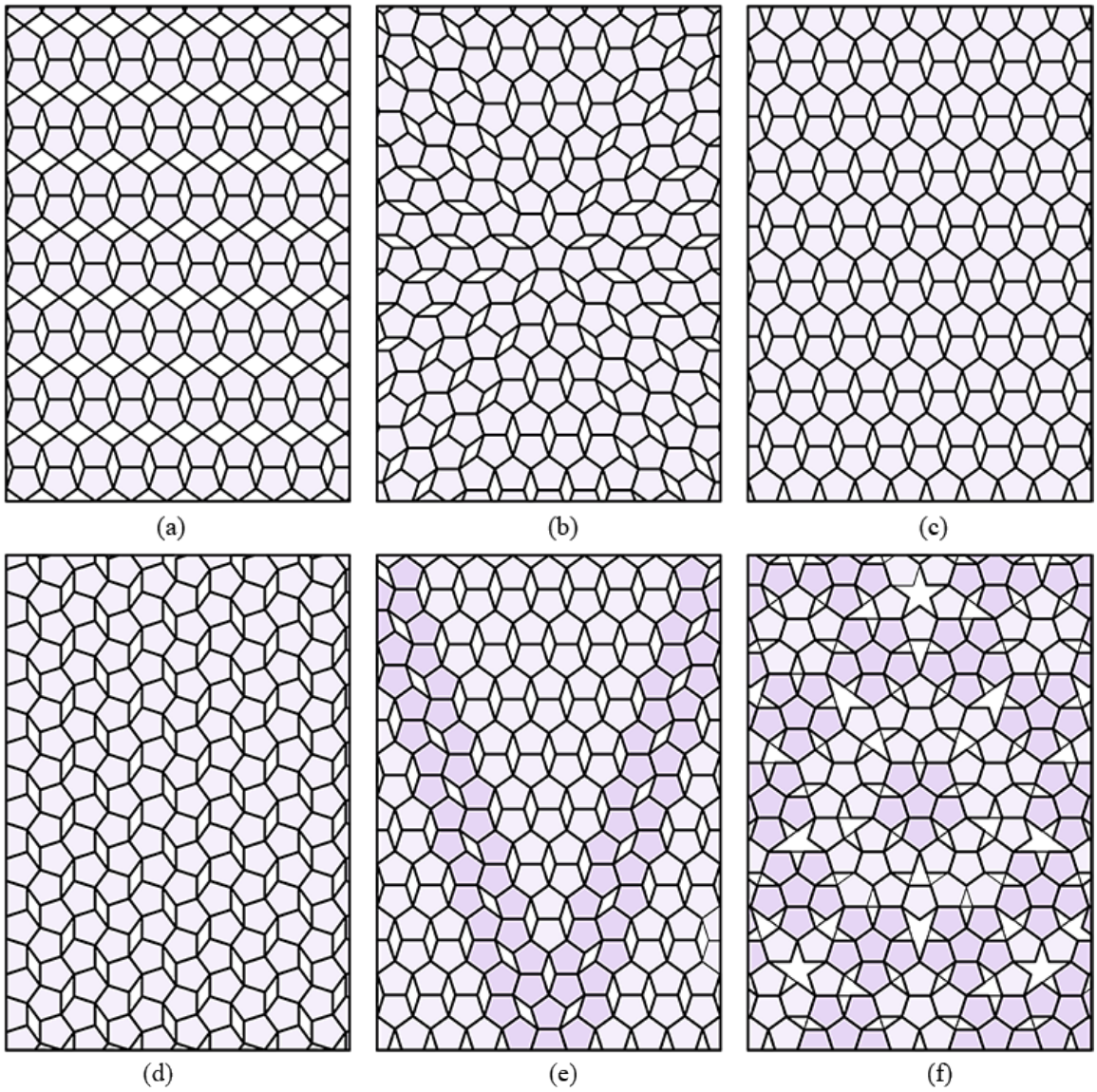


Figure 5.3 Patterns of the regular pentagon tiling. (a)-(b) Two patterns of pentagon tiling from Durer including a five-fold nucleus that is expanded by multiple twins of five-fold symmetry (reconstructed from [133]). (c)-(e) Three patterns of pentagon tiling in art proposed by Caris (reconstructed from [134]). (f) Penrose tiling with five-fold symmetry generated by attaching multiple groups of a pentagon to the initial one (reconstructed from [135]).

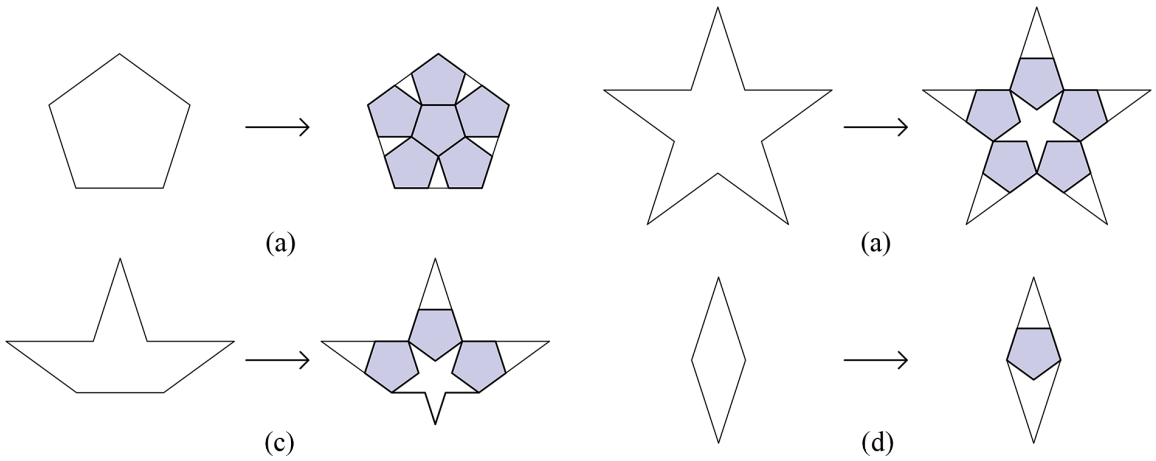


Figure 5.4 Substitution rules for Penrose tiling (reconstructed from [14]). (a) A pentagon is partially filled by 6 pentagons. (b) A pentacle is partially filled by 5 pentagons. (c) A half-pentacle is partially filled by 3 pentagons. (d) A rhombus is partially filled by 1 pentagon.

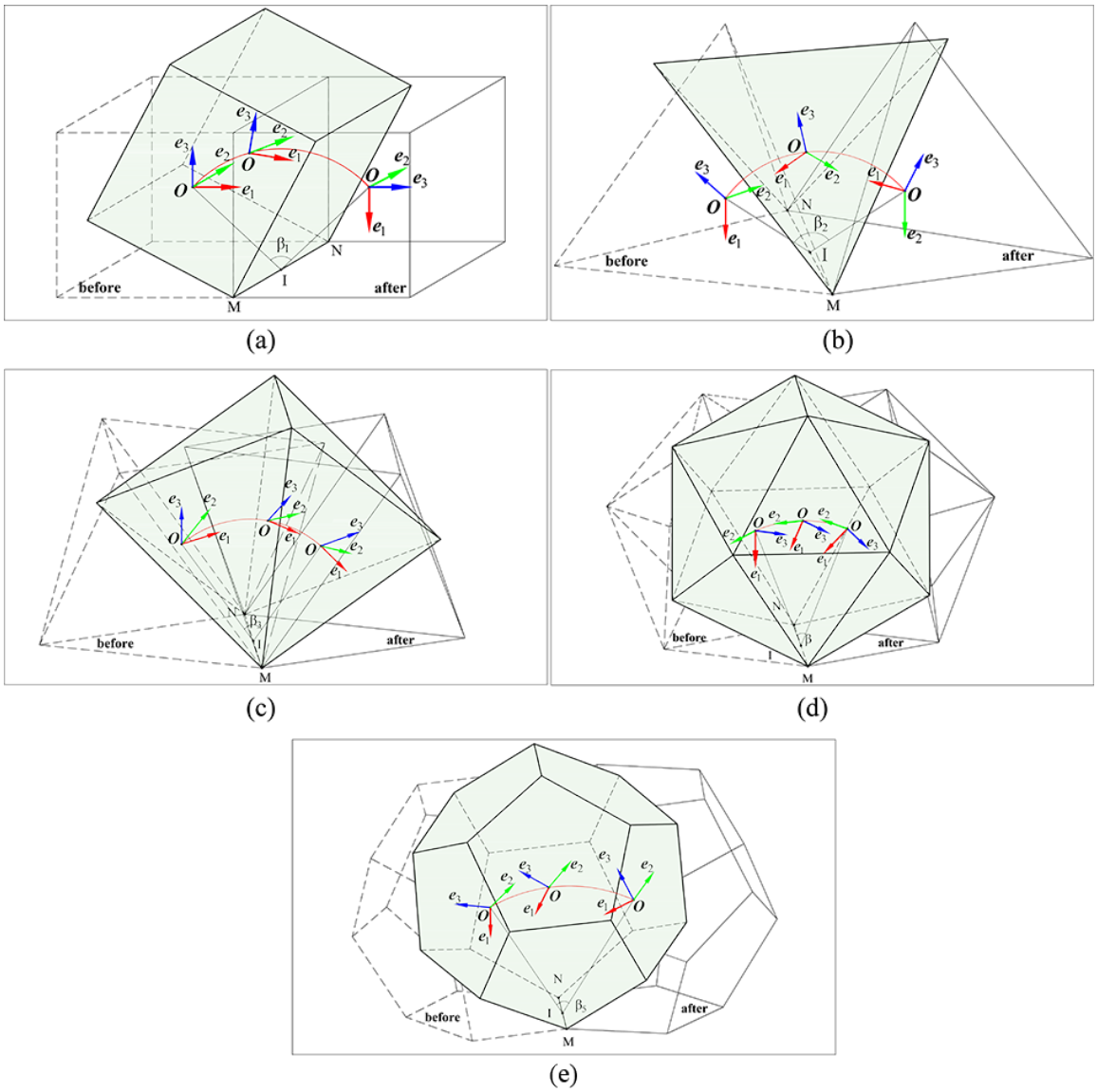


Figure 5.5 Configurations of the Platonic solids through edge-rolling. (a) Cube coordinates change from before state to the after state with β_1 rolling angle. (b)-(d) The respective of the rest Platonic solids with $\beta_2 - \beta_5$. $Oe_1e_2e_3$ is the coordinate of the polyhedron, which moves along the red curve which represents one step of rolling motion.

5.3 Result

This section introduces the results for the path planning of the Platonic solids on their respective grids. Fig 5.5 shows how edge-rolling changes the poses of the Platonic solids. Fig 5.11-5.12 shows the respective paths for the Platonic solids from their initial pose to their desired pose, where an additional Platonic solid shows an intermediate pose in-between.

5.3.1 Simulation Environment

We implemented the algorithm in MATLAB® on a PC with a 3.6 GHz Intel Core i7 processor. A space frame was fixed at the origin and a body frame was fixed on each Platonic solid; the plane was then discretized depending on the Platonic solid (Fig 5.5).

In this work, two cases studies are implemented for path planning fot Platonic solids through edge-rolling on the prescribed grids. The first case is to search path from a start pose, avoid obstacles and achive the goal pose. The second case runs with two steps. The first step is that the polyhedron roll directly from the start position to the goal position without considering the orientation. Then the second step is to implement the path planning BFS based algorithm from the last configuration to achieve the goal configuration. Fig. 5.6 shows an example of the triangular grid for

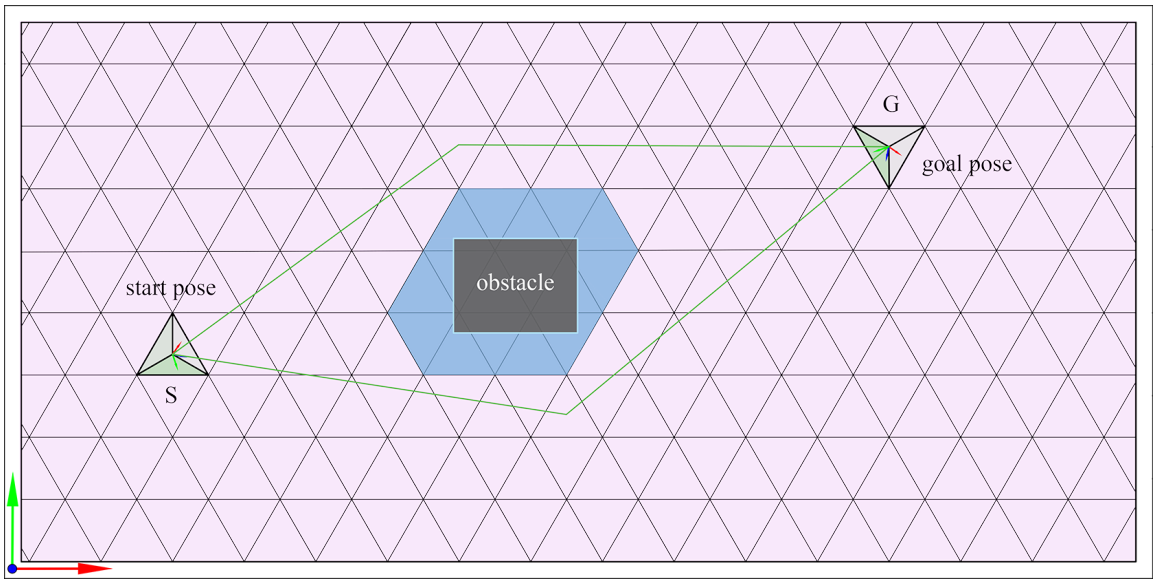


Figure 5.6 Simulation environment for rolling tetrahedron from a start pose to a goal pose while avoiding obstacle.

rolling tetrahedron, octahedron and icosahedron. The square grid is used for cube path rolling and penrose tiling is used for dodecahedron path rolling. The results of using the BFS based algorithm to find the rolling path for the Platonic solids are shown in the next subsections.

5.3.2 Obstacle Avoidance

In Fig. 5.6, a rolling polyhedron is at the starting configuration S and the target is to find a path to the goal configuration G while avoiding the polygonal obstacle, shown in black. A solution to this problem is to compute the euclidean shortest path from S to G that avoids the obstacle, as shown in green in the figure. However, rolling polyhedra is a nonholonomic system on a grid, which does not always follow the straight line as same as the rolling sphere with point contact constraint. Depending

on the polygon shape, a triangular can be used for tetrahedron model to found the boundary of the obstacle where the polyhedron does not contact to the obstacle Assuming that the reference point is the center of the polyhedron, the task to find a collision free path such that reference point moves from S to G in the case of shortest rolling path without considering the orientation.

The collision-free path cannot be guaranteed if the path planning algorithm does not account for the shape of the object. To avoid obstacle, the expansion of the obstacle region by setting it with the object shape is needed. After the setting step, the object can be represented as a single point with its coordinate system. The collision zone $\mathcal{O}_{obs}$ is the bounded region with light interior. Finding a collision-free path for the polyhedron corresponds to finding valid coordinate systems in $\mathcal{C}_{space}$ that connects the start configuration and the goal configuration. This approach is applied to the rest of Platonic solids.

5.3.3 Tetrahedron

A tetrahedron is constructed from 4 incident equilateral triangles, giving 4 vertices (Table 3.2) and an edge-rolling angle of $2\arctan(\sqrt{2})$ (Table 3.2) on the triangular grid. The symmetry of the tetrahedron limits its reachable poses, which can be seen as follows (Fig 5.7). We assume the surface S_{ct} of the tetrahedron as an initial configuration (bottom of Fig 5.7(a)-(b)) is in contact with the plane where the red arrow points down to the surface contact.

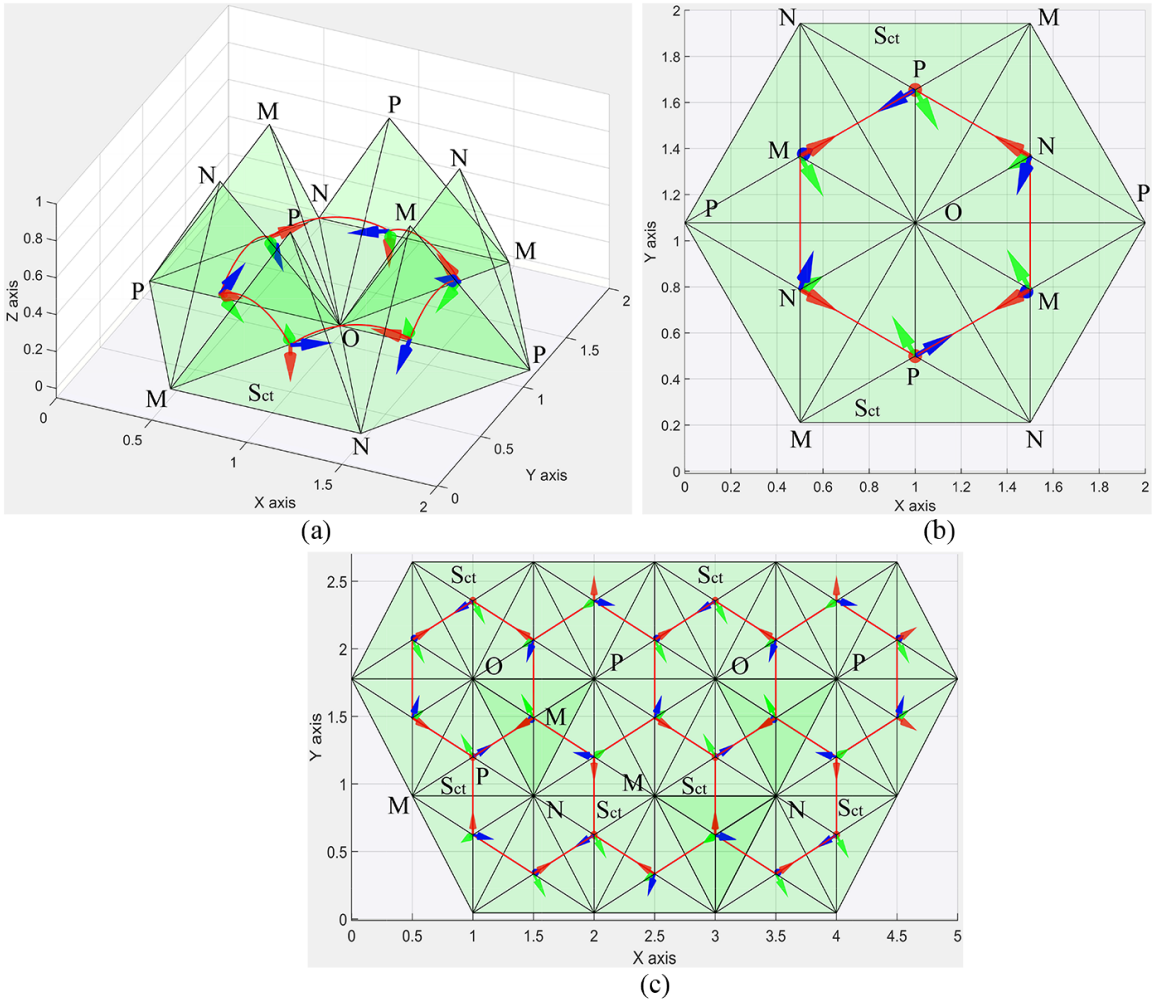


Figure 5.7 Symmetric properties of a tetrahedron. (a) A 3D view of edge-rolling 6 times around the vertex O where the red curve indicates the closed-path of rolling motion. (b) A top view of (a). The surface S_{ct} of the tetrahedron is in contact with the plane in different cell after sequential rolling through the edges of NO, PO, and MO to reach the same pose. (c) The tetrahedron reaches only one orientation for each cell through edge-rolling.

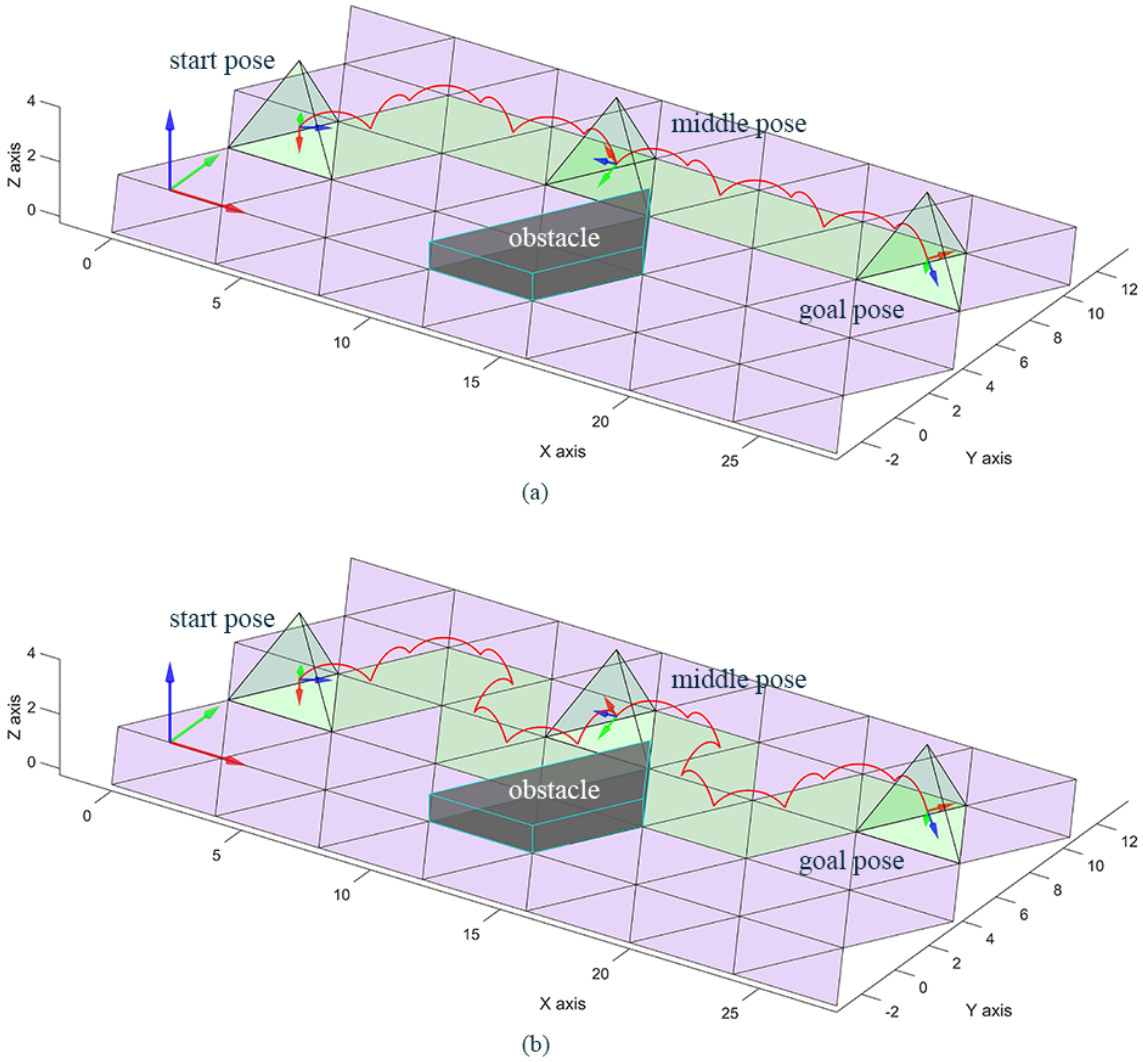


Figure 5.8 Tetrahedron rolling path in two different solutions. With the same scenarios, the number of rolling steps in (a) is less than the number of rolling steps in (a).

Because the tetrahedron has 3 incident faces at any vertex, edge-rolling along the edges NO, PO, and MO in sequence makes the face S_{ct} to be in contact with the plane again. Then, repeating this sequence of edge-rolling brings the tetrahedron back to the initial pose (more details in Fig 5.7(a)-(b)). As a result, the tetrahedron reaches the same pose after 6 times of edge-rolling around one vertex (Fig 5.7(b)-(c)) because the triangular grid which has 6 equilateral triangle shapes at any vertex. We conclude that only one pose can be reached for each cell starting from an initial

pose due to the high symmetry brought by the tetrahedron. The shortest path for the tetrahedron is shown in Fig 5.8.

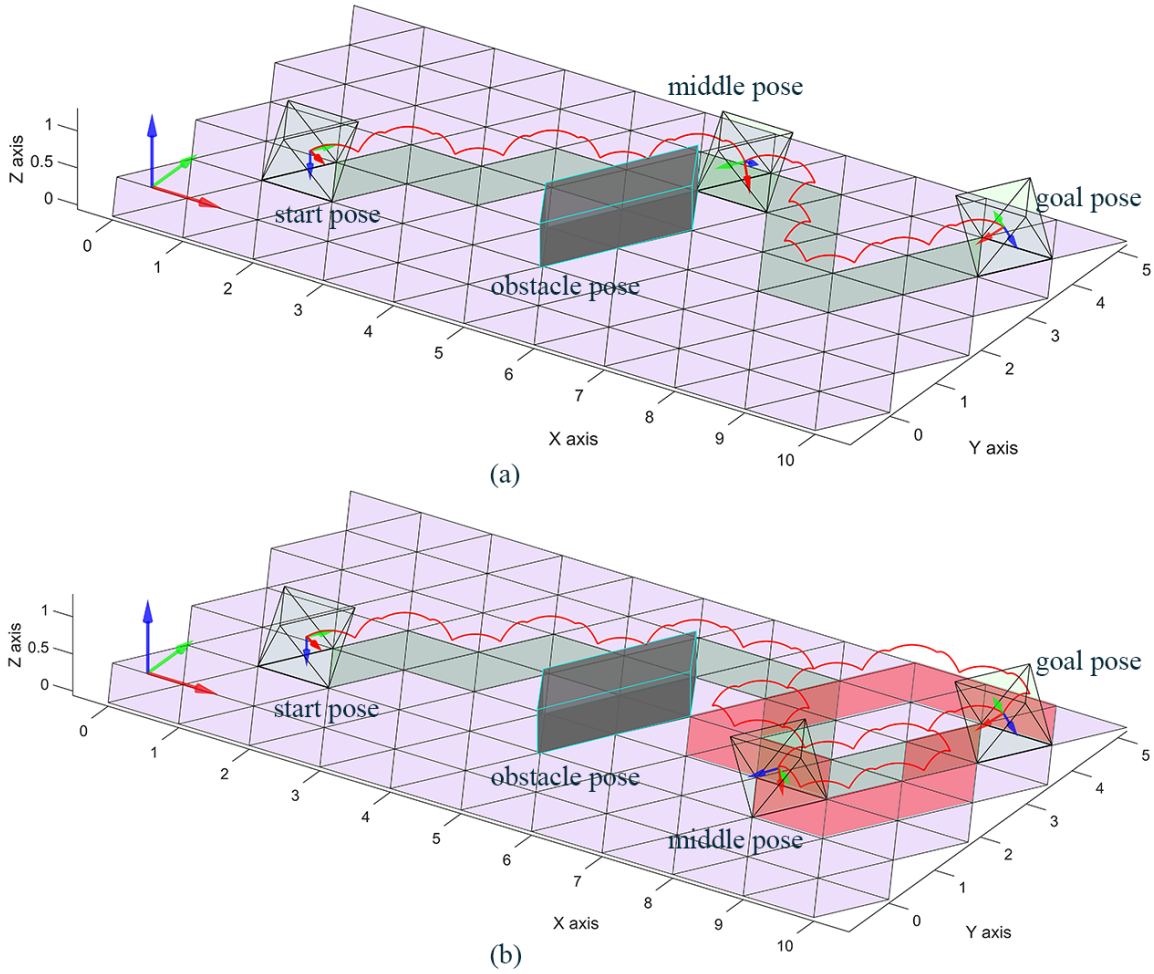


Figure 5.9 Octahedron rolling path while avoiding obstacles. (a) The octahedron has the rolling path indicated by the red curve and the light-green surface contacts. (b) There are two steps in this scenario, the light-green path represents the direct rolling path from the start pose to the goal position without considering the orientation. The red path represents the closed path to achieve the goal orientation at the same goal position.

5.3.4 Octahedron, Icosahedron, Cube, and Dodecahedron

On the other hand, the other four types of the Platonic solids can reach an arbitrary desired pose from an initial one because the increasing number of faces impose decreasing constraints. In these cases, each position is always reached by different orientation corresponding to various paths through due to the their symmetrical properties.

An octahedron is constructed from 8 incident equilateral triangles, giving 6 vertices (Table 3.1) and an edge-rolling angle of $\arccos(1/3)$ (Table 3.2). The shortest path for the octahedron is shown in Fig 5.9. An icosahedron is constructed from 20 incident equilateral triangles, giving 12 vertices (Table 3.1) and an edge-rolling angle of $\arccos(\sqrt{5}/3)$ (Table 3.2). The shortest path for the icosahedron is shown in Fig 5.10.

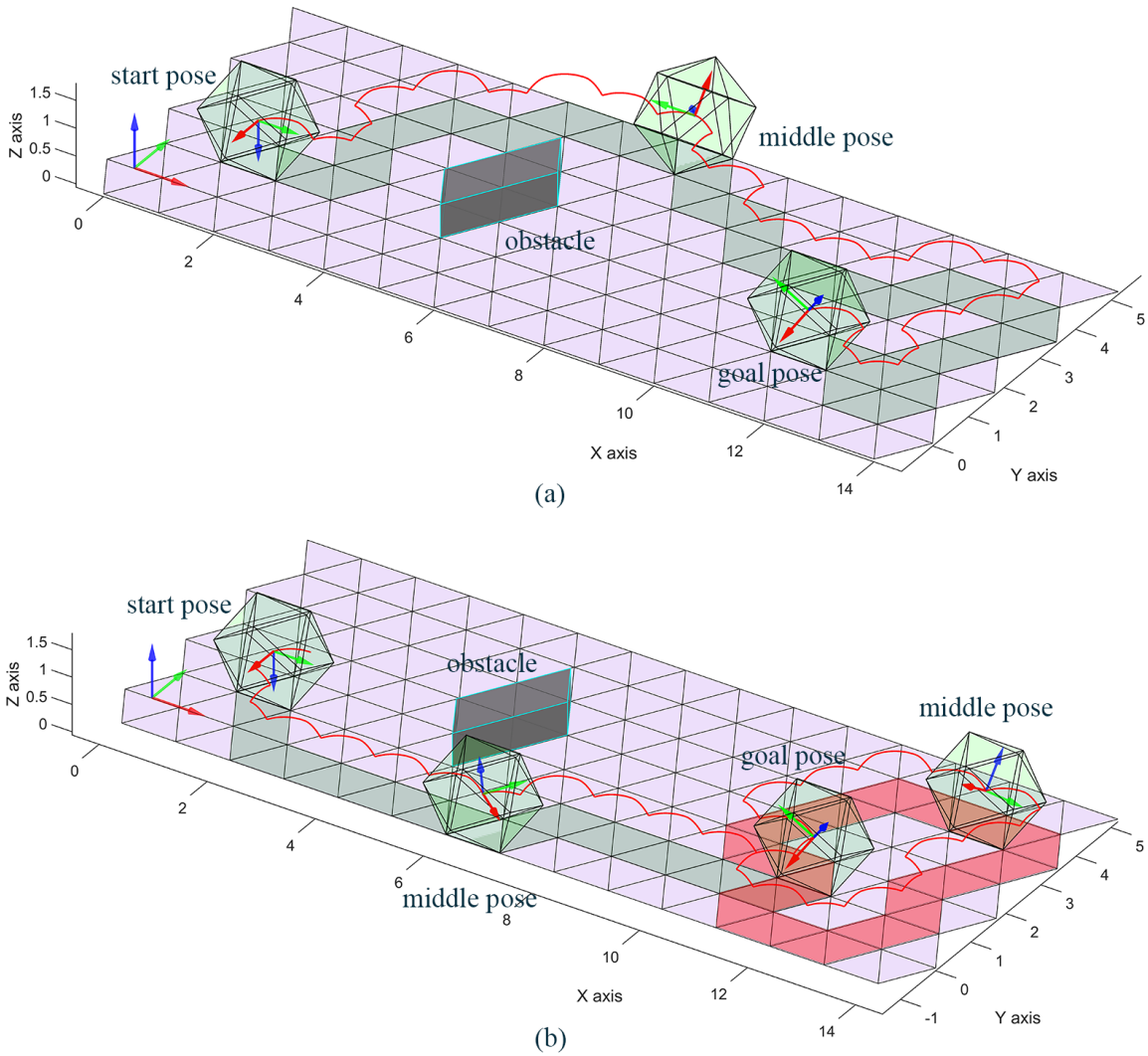


Figure 5.10 Icosahedron rolling path while avoiding obstacles. (a) The icosahedron has the rolling path indicated by the red curve and the light-green surface contacts. A middle pose is also displayed for visualisation (b) There are two steps in this scenario, the light-green path represents the direct rolling path from the start pose to the goal position without considering the orientation. The red path represents the closed path to achieve the goal orientation at the same goal position. It can be seen that two scenarios have the same start pose and goal pose but having two different paths

A cube is constructed from 6 incident squares, giving 8 vertices (Table 3.1) and an edge-rolling angle of $\pi/2$ (Table 3.2) on the square grid. The shortest path for the cube is shown in Fig 5.11. Finally, a dodecahedron is constructed from 12 incident regular pentagons, giving 20 vertices (Table 3.1) and an edge-rolling angle of $\arccos(\sqrt{5}/5)$ (Table 3.2). The shortest path for the dodecahedron is shown in Fig 5.12.

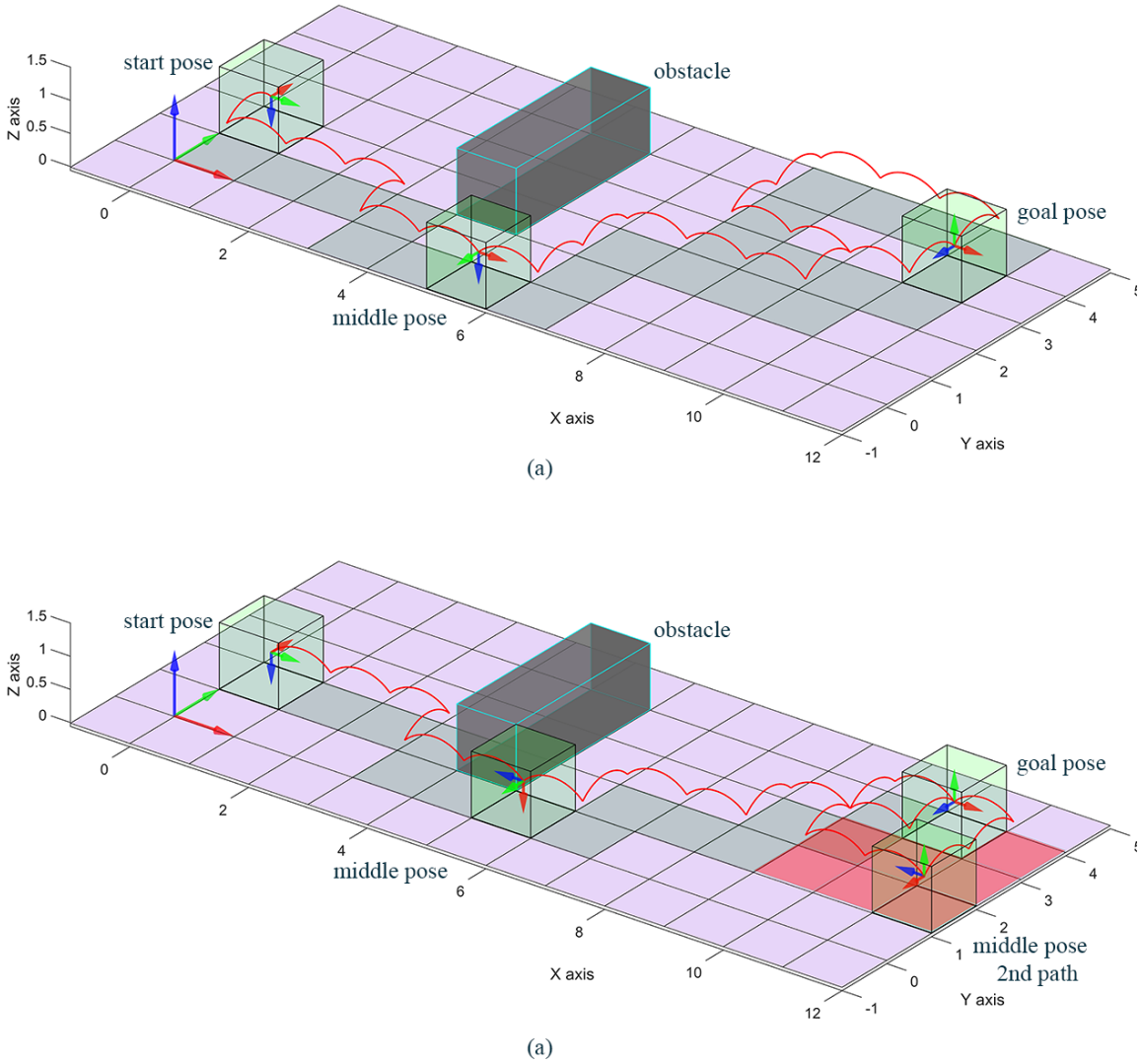


Figure 5.11 Cube rolling path with obstacle avoidance. (a) The cube has the rolling path indicated by the red curve and the light-green surface contacts. (b) There are two steps in this scenario, the light-green path represents the direct rolling path from the start pose to the goal position without considering the orientation. The red path represents the closed path to achieve the goal orientation at the same goal position.

Fig. 5.13 shows the results of the number of steps for each type of Platonic solids takes the rolling action while Fig. 5.14 shows the searching time for the path planning algorithm to find the path in the two case studies of Platonic solids through edge-rolling contact. It can be seen clearly that the icosahedron takes more steps to achive the goal while the tetrahedron requires less steps of rolling due to the its

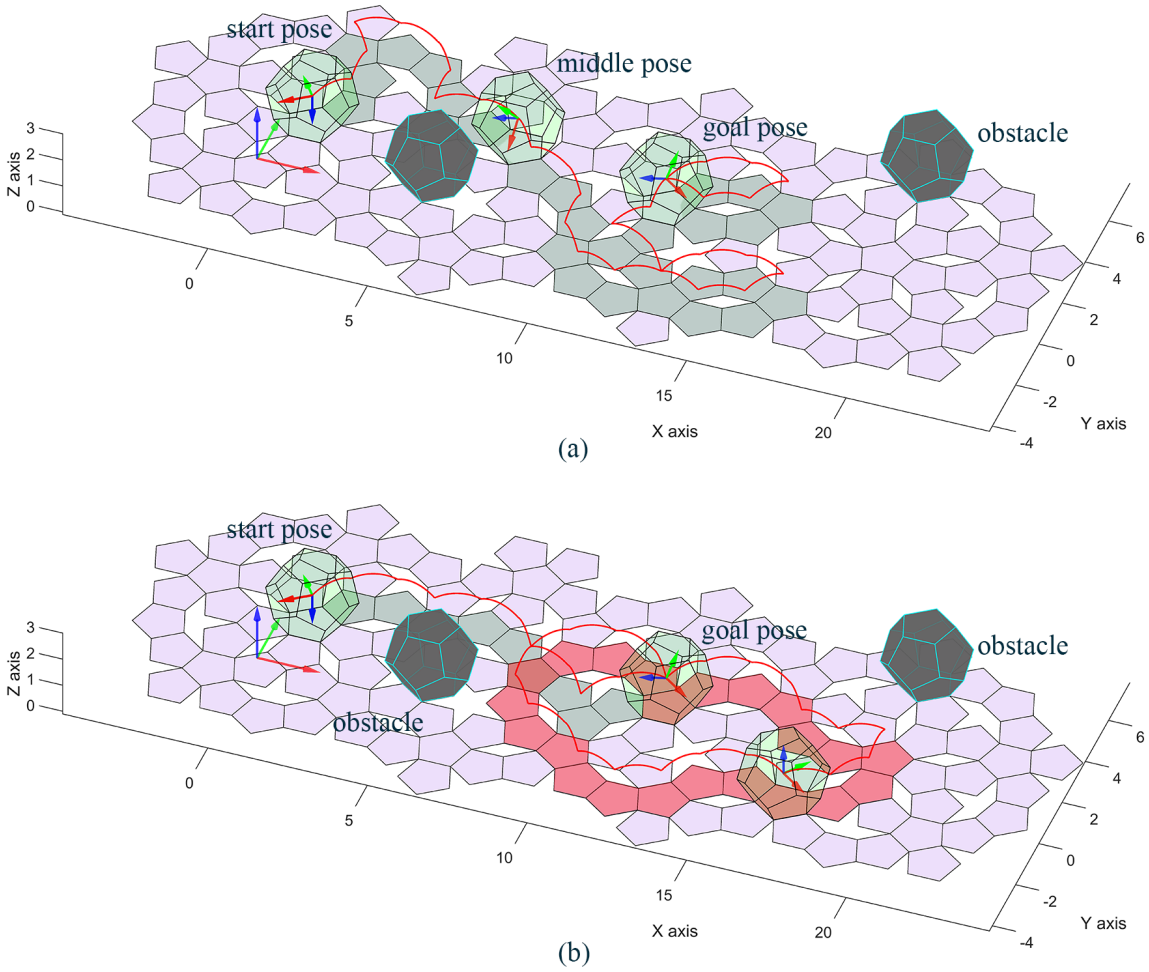


Figure 5.12 Dodecahedron rolling path with obstacles avoidance. (a) The dodecahedron has the rolling path indicated by the red curve and the light-green surface contacts. (b) There are two steps in this scenario, the light-green path represents the direct rolling path from the start pose to the goal position without considering the orientation. The red path represents the closed path to achieve the goal orientation at the same goal position.

symetrical properties. In the other hand, the dodecahedron and icosahedron cost longer searching time to compare to the rest of the Platonic solids to find the path due to their complex geometry.

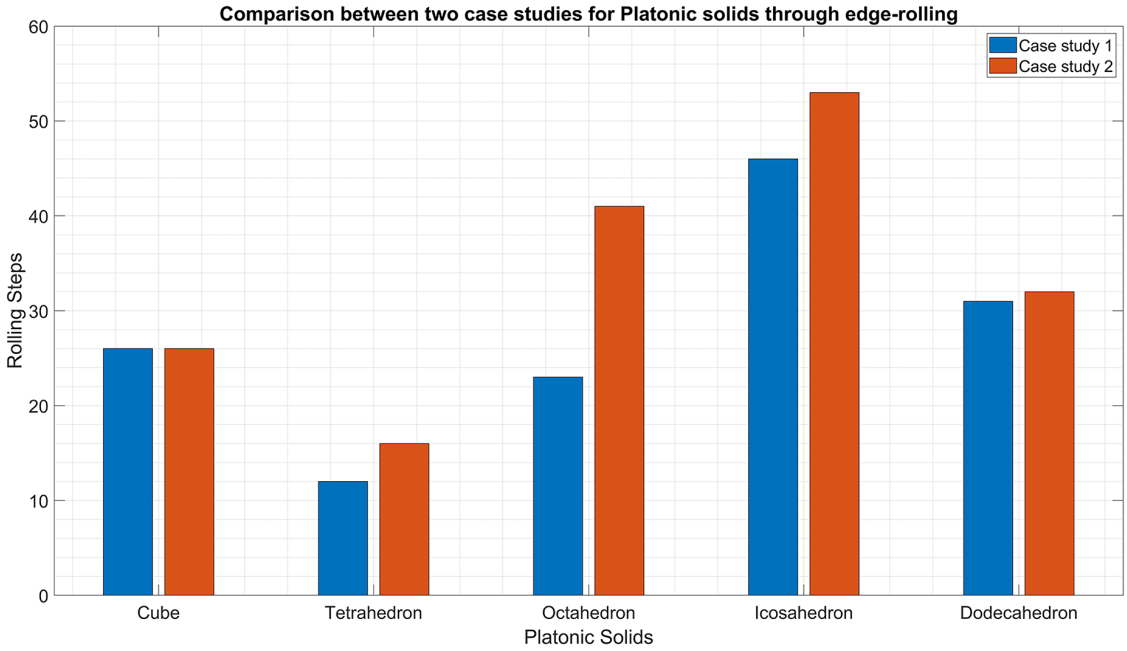


Figure 5.13 The comparison of the rolling steps of the Platonic solids between two senarios.

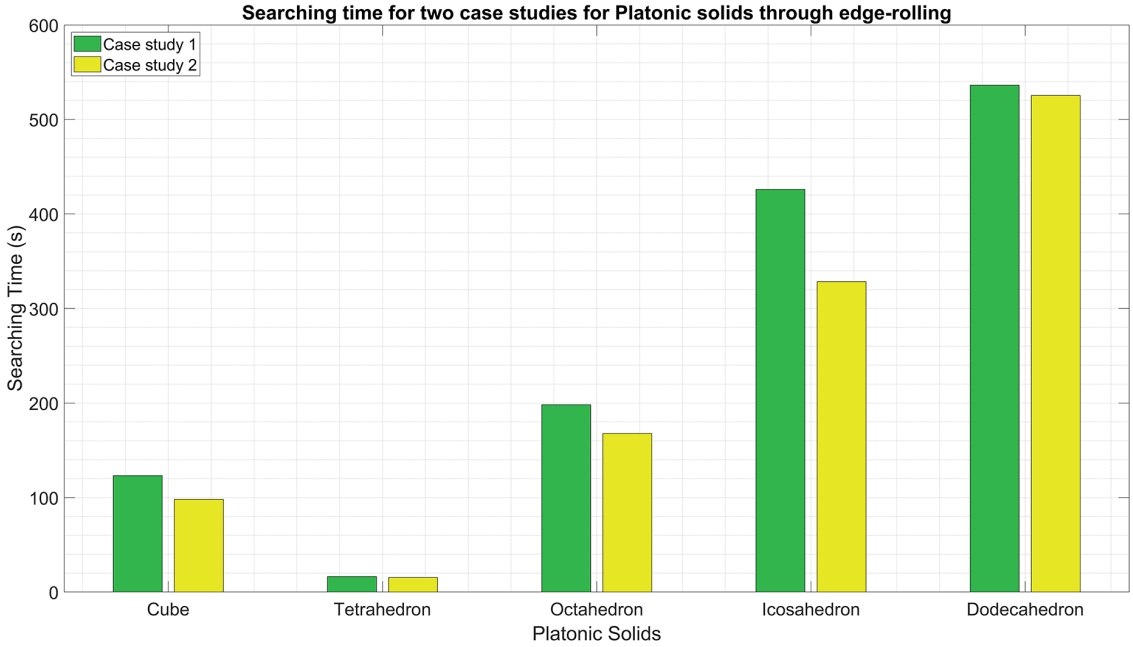


Figure 5.14 The comparison of the searching time of the Platonic solids between two senarios.

5.4 Conclusion

The study propose a path-planning algorithm for the Platonic solids from an initial pose to a desired pose on a plane by edge-rolling. It is straightforward to tile a plane with equilateral triangles for the tetrahedron, octahedron, and icosahedron and with squares for the cube, but there are a variety of regular pentagon tiling patterns, which all leave symmetric gaps in the plane. The study chose Penrose tiling because the rhombi gaps exhibit five-fold symmetry, which facilitates the proposed algorithm. While the cube, octahedron, icosahedron, and dodecahedron can reach an arbitrary desired pose from an initial one, the tetrahedron can only reach one orientation for a cell due to the high symmetry brought by the tetrahedron and triangular grid. This study successfully solved the path-planning problem of the Platonic solids by edge-rolling without obstacles on prescribed grids. The study are currently investigating the possible extension of the optimal searching algorithms to more general convex solids and the optimal searching algorithm.

Chapter 6: Path planning RRT based algorithm for Truncated Icosahedron [1]

The study of convex polyhedra has long been attractive to many researchers in mathematics, science and art. Most of the previous studies focused on geometrical properties, spatial, unfolding polyhedron, polyhedral intersection and pattern combination. An early discussion of regular polyhedra concerned the five Platonic solids - tetrahedron, cube, octahedron, icosahedron and dodecahedron mentioned by Plato in Timaeus. Rolling polyhedra has been suggested, but not validated in any path planning algorithm, except for solving the rolling-cube puzzles for cubic dice. The aim of this study is to develop discrete path planning algorithms for rolling convex polyhedra to achieve a random configuration from an initial configuration on a plane. The method applies to the truncated icosahedron - an Archimedean solid shaped by regular hexagonal and pentagonal faces. The path planning method based on the rapidly-exploring random tree (RRT) algorithm is improved to find a closed path for rolling truncated icosahedron from an initial configuration to a random goal configuration on a plane with multiple obstacle avoidance. The outcome of this study is to suggest improvement path planning algorithms, by demonstrating significant results

[1] This chapter will be submitted as a journal paper: N.T. Lam, I. Howard, L. Cui (2022) "Discrete Path Planning Based RRT Algorithm for Truncated Icosahedron through Edge-Rolling on a Plane". *Journal of IEEE Intelligent Systems* (to be submitted).

in finding the shortest paths for the truncated icosahedron through edge rolling to achieve a target, and that the result can be applied to any convex polyhedron. The applications of this study can be used in dexterous robotic in-hand manipulation, robotic rolling or game applications.

6.1 Introduction

In previous works it has been noted that sampling-based algorithms have been used to solve path planning problems to find feasible paths that connect the starting point to the goal point with obstacle avoidance. Many works focused on addressing the problem of manipulation planning for robotic tasks by investigating minimally complex hardware systems [139, 140]. In the robotics literature, the different contexts of geometric, mechanical, and control of dexterous manipulation which are not trivial to change arbitrarily the position and orientation of the manipulated object by rolling contact were presented [22]. The approach in [141] reduced the problem of planning for general surfaces but the convergence rate is slow without considering the possible presence of obstacles. Accurately resolving the contacts between polyhedra and surface such as cube manipulation by fingertips is difficult and computationally expensive compared to sphere objects [142]. The graspless manipulation of the polyhedral parts was discussed in [68, 17, 18]. These works focused on the theoretical analysis of the set of reaching polyhedron configurations. However, there was a lack of complete planning and analysing of different geometry in discrete mathematics for

the rolling motion of polyhedral parts on regular surfaces. The path planning for manipulation of polyhedral parts in [67] is considered a nonholonomic system among obstacles and the early general approach to this problem was presented by [143]. This study introduced a general property of the topology of exact planning in constrained spaces. Although there exists a study by Lam *et al.* [144] regarding path planning for the Platonic solids on prescribed grids by edge-rolling, to the best of our knowledge, no study has been conducted on the rolling path generation of the truncated icosahedron on a plane while avoiding obstacles collision. This study aims to establish a path planning method-based RRT algorithm to find an optimal path for rolling a truncated icosahedron which is an Archimedean solid to achieve a goal configuration on a plane. The properties of truncated icosahedron which is associated with soccer balls and consists of hexagons and pentagons represent the general convex polyhedra. The path planning method is developed from a basic RRT algorithm [101] to tackle the nonholonomy of rolling convex polygon by the discrete path-finding on a plane.

The main contributions of this work are as follows:

- The geometrical properties of truncated icosahedron which is created from the truncation of an icosahedron - one of the five geometric solids of Platonic, are analysed with rolling contact.
- Given the information of the plane and obstacles, the closest paths are found from a start configuration to a goal configuration. The method is developed based on the RRT path planning technique, which is used for truncated icosa-

hedron through edge-rolling without obstacle collision on a plane.

- The method can be developed for rolling general convex polyhedra on deformable surfaces and applied to in-hand manipulation robotics.

The rest of this chapter is organized as follows: Section 6.2 briefly presents a review of the related literature on the path planning algorithms for rolling polyhedra. Section 6.3 analyses the details of geometrical properties of the truncated icosahedron in terms of edge-rolling contact. Section 6.4 demonstrates simulation results of path planning algorithm based on the RRT method for truncated icosahedron though edge-rolling contact and Section 6.5 concludes this study and drives some directions for future work.

6.2 Path Planning for Polyhedra

Literature reviews on path planning algorithms have been done significantly in Chapter 3. However, most of the previous path planning techniques applied in the autonomous system are related to the dynamics and trajectory scope. Chapter 4 reviews the relevant literature regarding RRT-based algorithm for path planning. The method takes into account solving problems with obstacles and nonholonomic constraints. This algorithm has been widely applied in sampling-based path planning for robotics [130]. However, there is no previous application of this algorithm to solve the problem of path-finding for the polyhedra.

In terms of polyhedra manipulation, in the early 18th century, the conception of differential geometry has been discussed in classical rational mechanics. The system was named the non-holonomic system which is opposed to the nonholonomic system. The survey [145] introduced brief historical references and several examples of the systems. An example of the idea for working on the nonholonomic system of the rolling surfaces constraint so that the design of a dexterous hand was introduced in [146]. This work was developed from the previous study of Lin and Canny [67] in the results of arbitrarily changing the position and orientation of an object with respect to a regular surface. The experiment of this study was to demonstrate the rolling of a sphere on a plane from an initial position and then bringing it back to this position with a different orientation.

Chaplygin [147, 148] provides a sphere rolling on a plane without spinning and sliding. The studies illustrated integrals and equations of motion of rolling balls in different systems. Li *et al.* [140] and Cole *et al.* [149] discussed the dexterous manipulation through rolling without slipping two objects. The nonholonomic constraint such as the relationship between the velocity of two manifolds and their configuration is derived to minimize the coordinates parameterization. These main technical contributions are the derivation of a complete model of the rolling contact. However, the approach has been focused on the dynamics system.

Marigo and Bicchi [38] discussed the planning algorithms for chained-form systems in free space. This method provides the controllability aspect by giving a complete

reachable manifold for the two bodies and proposes a constructive controllability technique for rolling motion planning for dexterous robot hands. Marigo *et al.* [68] proposed an attempt of path finding for an octahedron edge-rolling on a plane from a start point to a goal point, which, unfortunately, failed due to errors propagating from the algorithm.

Erdmann *et al.* [18] introduced the problem of rolling a cubic dice with edge-contact of the two movable parallel plates working as a robotic end-effector in the terms of graspless manipulation. However, the method to generate the desired path has not been discussed in the study. In the mathematics aspect, the rolling-cube puzzles were solved by finding the Hamilton path for grid graphs, which is called an NP-complete problem [69]. Nevertheless, this work focused on the simple problem of how to roll cubic dice on a board consisting of labelled and white cells, which is hardly extended to other complex polyhedra.

In this work, we propose a path planning algorithm using tree exploration for the truncated icosahedron starting from an initial configuration to the desired configuration by edge-rolling on a plane while avoiding obstacles.

6.3 Truncated Icosahedron

This section contains a detailed analysis of a truncated icosahedron. The rolling contact of the polygon and its unfolding problems then are provided, based on the mathematic literature review.

6.3.1 Geometry Properties

Platonic Solids are regular convex polyhedra, established by intersecting of equivalent faces. A truncated icosahedron is one of the 13 Archimedean solids, which is a convex polyhedron. One difference in structural properties between the truncated icosahedron and Platonic solids is that the truncated icosahedron is made by nonintersecting two or more different types of regular convex polygons [33]. A truncated icosahedron can be obtained from Platonic Solids in a variety of ways by truncating an icosahedron as an example which is shown in Fig. 6.1. All 12 vertices of the icosahedron are truncated in order to generate new truncated icosahedron edges with one-third of the length of the icosahedron's edge. This truncated step creates the 32-faced Archimedean solid comprising 12 pentagon faces and 20 regular hexagon faces (consisting of a total of 60 vertices and 90 edges), and the radius of the sphere is also one-third of the polyhedron side.

A truncated icosahedron is the model represented in the construction of soccer balls.

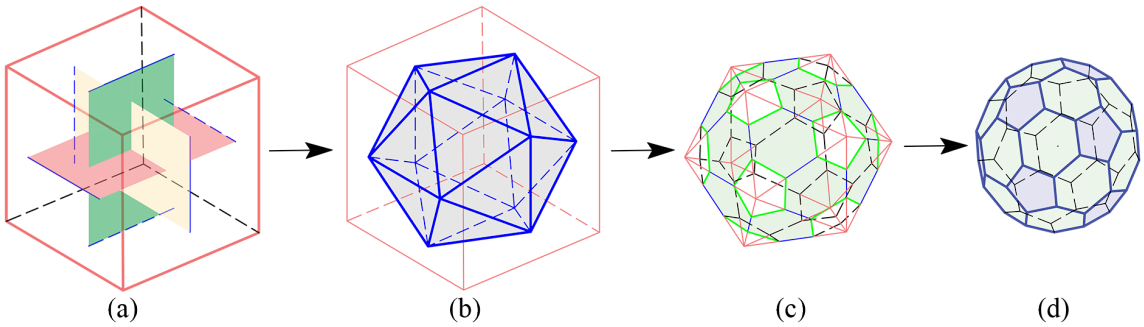


Figure 6.1 Generating a truncated icosahedron from a cube. (a) Picking the edges in the middle of the cube with the intersection of the three perpendicular planes at the centre of the cube. (b) Connecting the vertices from each edge so that creating equilateral triangular surfaces, the icosahedron is then constructed. (c) Truncating each vertex of the icosahedron generates a truncated icosahedron, as shown in (d).

The model also appears in different structures in research such as the lens configuration of the Fat-Man atomic bomb [150] or the $C\_(60)$ pure carbon structure of Buckminsterfullerene [151]. The truncated icosahedron is highly symmetric and has lots of roughly evenly spaced faces. The symmetry group is transitive on the hexagons, the pentagons, the hexagon-hexagon edges, the hexagon-pentagon edges and the corners. This requires only the calculation of the shock wave geometry for a pentagon, a hexagon, and two kinds of edges between them, which makes the lenses substantially smaller, incorporating less sharp angles. As can be seen in Fig. 6.2, the truncated icosahedron is projected onto its circumscribed sphere to generate a soccer ball. The simple way to achieve this is to project each polygon of the truncated icosahedron onto the sphere through a parametric equation (Equation 6.1), the next step is to project the generated arc onto the $2D$ plane.

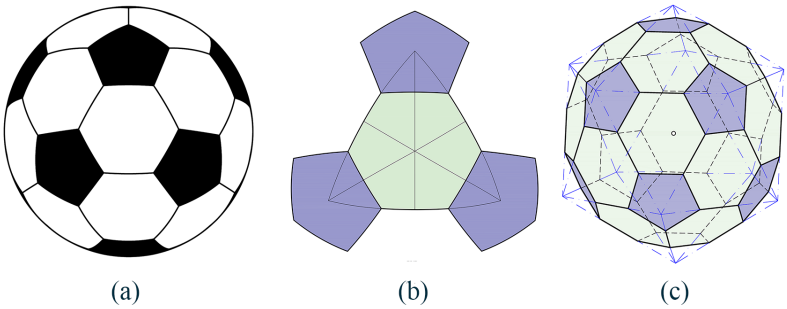


Figure 6.2 The soccer ball (a) is obtained by the projection of the truncated icosahedron onto the circumscribed sphere (b) through parametric equations.

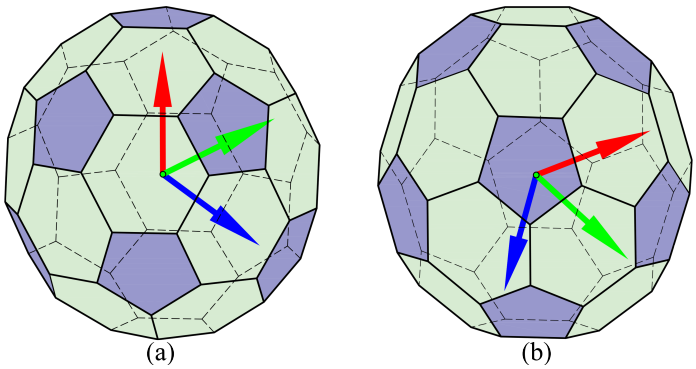


Figure 6.3 Different orientations of the truncated icosahedron correspond to their coordinate systems. (a) Pentagon-face contact, (b) Hexagon-face contact.

The technique can be expressed by giving an edge E_1E_2 with $3D$ parameters $E_1 = (x_{E_1}, y_{E_1}, z_{E_1})$ and $E_2 = (x_{E_2}, y_{E_2}, z_{E_2})$. The parametric equation of the edge is $\mathcal{V}(t)$.

$$\mathcal{V}(t) : \begin{cases} \mathcal{X}(t) = x_{E_1} + (x_{E_1} - x_{E_2})t \\ \mathcal{Y}(t) = y_{E_1} + (y_{E_1} - y_{E_2})t, 0 \leq t \leq 1 \\ \mathcal{Z}(t) = z_{E_1} + (z_{E_1} - z_{E_2})t \end{cases} \tag{6.1}$$

The norm of $\mathcal{V}(t)$ then can be obtained by: $\|\mathcal{V}(t)\| = \sqrt{\mathcal{X}^2(t) + \mathcal{Y}^2(t) + \mathcal{Z}^2(t)}$. The projection of the edge E_1E_2 on the unit sphere satisfies a parametric equation $\mathcal{P}_{\mathcal{V}(t)} = \mathcal{V}/\|\mathcal{V}(t)\|$.

In the static condition, the truncated icosahedron has two types of polygon contacts to a surface: hexagon-based contact and pentagon-based contact, as shown in Fig. 6.3. Truncated icosahedron consists of a regular pentagon and a hexagon as described above. One hexagon is surrounded alternating by 3 pentagons and 3 hexagons while each pentagon is surrounded by 5 hexagons. The study is on rolling through edge contact, then the property of the edge which joins with other two edges meeting at each vertex- a common line of an intersection between a pentagon and a hexagon and an intersection between a pentagon and hexagon - is considered to determine the edge contact.

A specific geometrical property of a truncated icosahedron is to follow the twelve pentagons theory. If every polygon face of the polyhedron model is a hexagon or a pentagon, and each vertex connects to the three intersection edges, the polyhedron has exactly twelve pentagonal faces. The theorem is proved by Euler's theory. Let $\mathcal{P}$ denote a polyhedron rolling on a plane Π, and

- $\mathcal{F} = \{F_1, ..., F_r\}$ the set of its faces.
- $\mathcal{E} = \{E_1, ..., E_k\}$ the set of its edges;
- $\mathcal{V} = \{V_1, ..., V_h\}$ the set of its vertices;

Then, $n_{F_{hexa}}$ and $n_{F_{penta}}$ are denoted as the number of hexagonal and pentagonal faces respectively. The total number of polygon faces is $\mathcal{F} = n_{F_{hexa}} + n_{F_{penta}}$. The number of edges is counted by $\mathcal{E} = (5/2)n_{F_{hexa}} + 3n_{F_{penta}}$. Then the number of vertices is

determined through the edges as every vertex is the point intersecting three edges $\mathcal{V} = (2/3)\mathcal{E}$. Based on Euler's theory $\mathcal{V} - \mathcal{E} + \mathcal{F} = 2$, the relationship between faces and edges is $3\mathcal{F} - \mathcal{E} = 6$. The final result of the pentagon faces $n_{F_{penta}} = 12$ by inserting $\mathcal{F}$. For example, in a golf ball, the family of ball shapes including the truncated icosahedron is constructed from 232 polygonal faces and it also has 12 pentagonal faces.

6.3.2 Rolling Contact

Nonholonomic Constraint

Given two objects in single-point contact, if the contact is maintained, three roll-non-slide configurations are considered in this study (Fig. 6.4). The surface contact can be treated as polygon vertices contacts (six vertices for the hexagon and five vertices for the pentagon) while two vertices contact represents a line contact. Let $\kappa = (c_1, c_2)$ to denotes the contact configuration between the truncated icosahedron $\mathcal{P}$ and the plane π. A distance function d_κ between two objects that are positive when the truncated icosahedron does not contact the plane, zero when they are contacting and negative when they are in penetration. Kinematic modelling including friction or soft contact, slipping and velocity is not considered in this thesis.

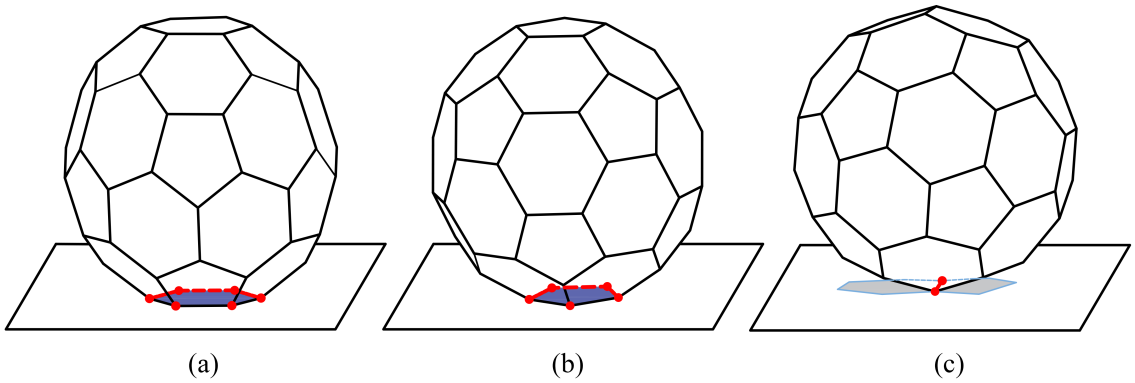


Figure 6.4 Different types of truncated icosahedron rolling contacts to the surface. (a) A surface contact can be treated as six-point contacts for hexagon faces. (b) A surface contact can be treated as five-point contacts for pentagon faces. (c) A line contact can be treated as two-point contact for any polygon face.

There are no constraints on the rolling motion of the polygon when $d_\kappa > 0$ or it has six degrees of freedom. With $d_\kappa = 0$ in this study, the roll-non-slide motion is called the nonholonomic constraint. The required condition to maintain the contact is $\dot{d}_\kappa = 0$. There are two constraints here: the constraint of the surface contact maintained while the polygon is stable and the constraint of the edge contact when the polygon is rolling to achieve a new configuration.

Unfolding Polyhedron

In this thesis, path planning for truncated icosahedron rolling considers the surface contact between the polyhedron model and the surface, while the line contact is for taking rolling action. Unfolding the polyhedron onto the surface gives basic contact surfaces that the polyhedron can roll through all surfaces of the model. Two general techniques named the `source unfolding` [152] and the `star unfolding` [153] are used to an unfold convex polygon model from a surface $\mathcal{F}$ to a non-overlapping polygon in

a plane. Both techniques are studied based on the shortest paths on the polyhedron surface with respect to a point $x \in \mathcal{F}$. The `source unfolding` method is obtained by cutting the `cut locus` while various paths to x can be generated from locus of points. The `star unfolding` is made by cutting a shortest path from every vertex of the convex polyhedron to the point x. However, the simple unfolding method is to cut along the edge of the polyhedron model to keep the original shape of each polygon face (hexagons and pentagons for the truncated icosahedron model).

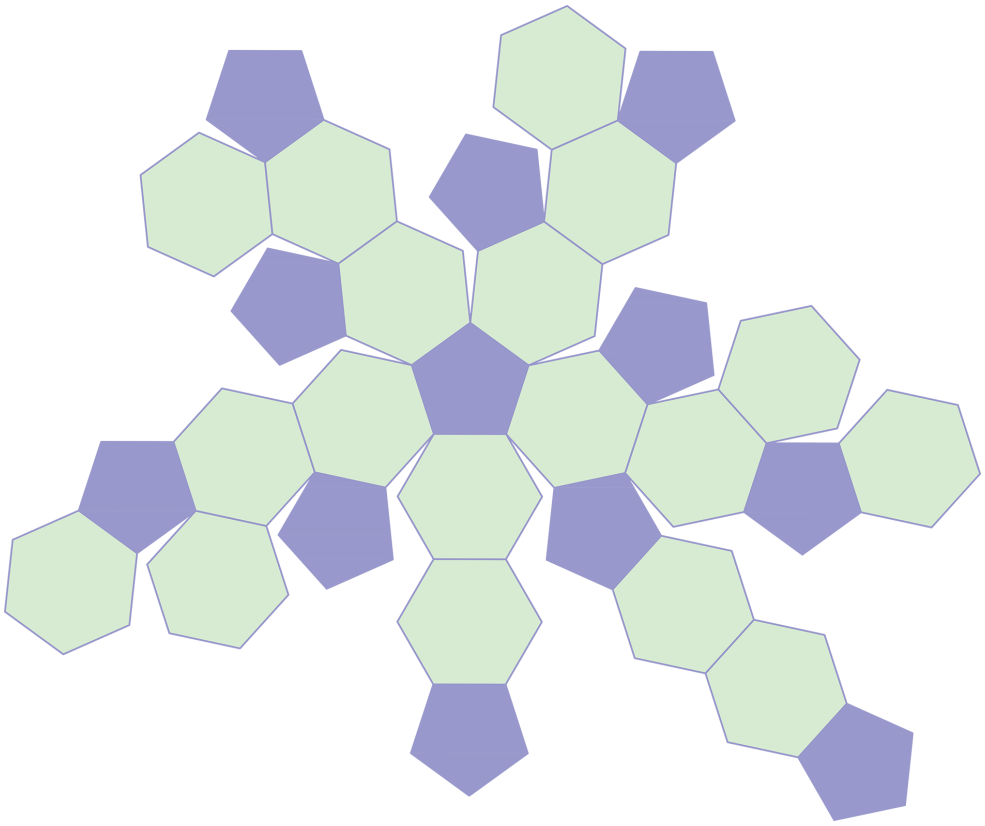


Figure 6.5 Two in several symmetrical projections of an unfolded truncated icosahedron on a plane. (a) Horizontal unfolding with fixed hexagon polygons and interchangeable pentagon polygons. (b) Center spreading-like flowers of unfolding truncated icosahedron with a pentagon face in the centre.

Unfolding truncated icosahedron can generate several symmetrical projections onto a plane that are illustrated in Fig. 6.5. The figure shows two patterns from several tiling truncated icosahedrons onto a surface. The pentagons (light purple) can be inter-cutting the boundary of the plattened polygon along its edges and can make a polygon model from a sheet of paper.

Rolling Directions

Although the truncated icosahedron can roll itself on the unfolding polygon, the actually rolling contact to reach random goal configurations is different and complex due to the connection gaps between hexagons and pentagons. As can be seen from Fig. 6.6, if the polygon contact is a hexagon, there are two possible ways of rolling: rolling from hexagon-based surface contact to hexagon faces (green arrows) and to pentagon faces (black arrows) (Fig. 6.6a) while the pentagon-based surface contact has only one type of rolling, from pentagon face to hexagon face (brown arrows) shown in Fig. 6.6b.

The $3D$ visualizations of both cases of the rolling are represented in Fig. 6.7(a-c) for rolling a truncated icosahedron from a hexa-based face to a hexagon face and the Fig. 6.7(d-f) for rolling a truncated icosahedron from hexa-based face to pentagon face. Fig. 6.7 indicates one step of rolling from a hexagon-based face to a hexagon face. The rotation angle is calculated by $\theta_{66} = \pi - \arccos(-\sqrt{5}/3) \simeq 41.81$ (degree). The calculation of rolling angle for the case of hexagon-based face to pentagon face

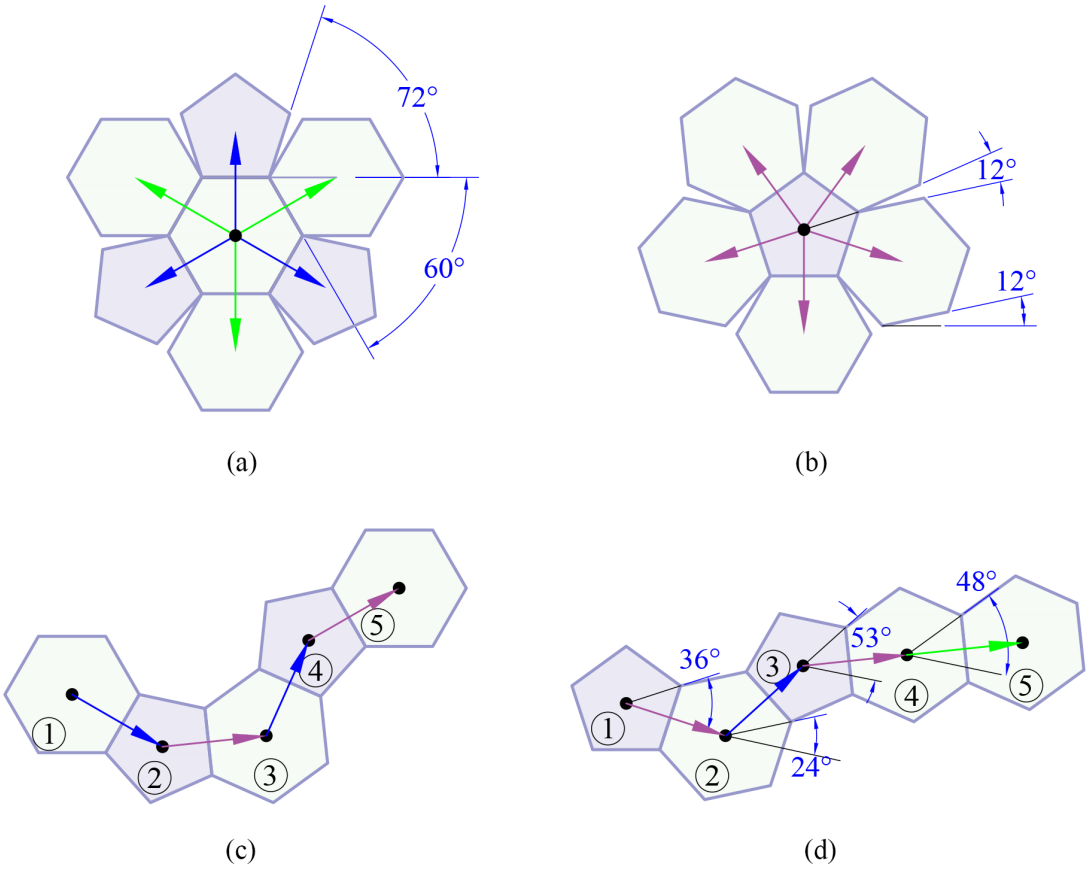


Figure 6.6 Two types of polygon contacts from a truncated icosahedron: (a) Hexagon surface-based contact and (b) Pentagon surface-based contact. (c)&(d) Examples of next surface-based contacts are represented in different rolling directions.

is more complex as $\theta_{65} = \pi - (\arctan\sqrt{\frac{7+3\sqrt{5}}{2}} + \arctan((\tan(\frac{\pi}{5}))\sqrt{\frac{125+41\sqrt{5}}{10}}))$ which is approximately of 37.37 (degree) (Fig. 6.7). Fig. 3.10 shows in detail the dihedral angles between two pairs: hexagon-hexagon faces and hexagon-pentagon faces.

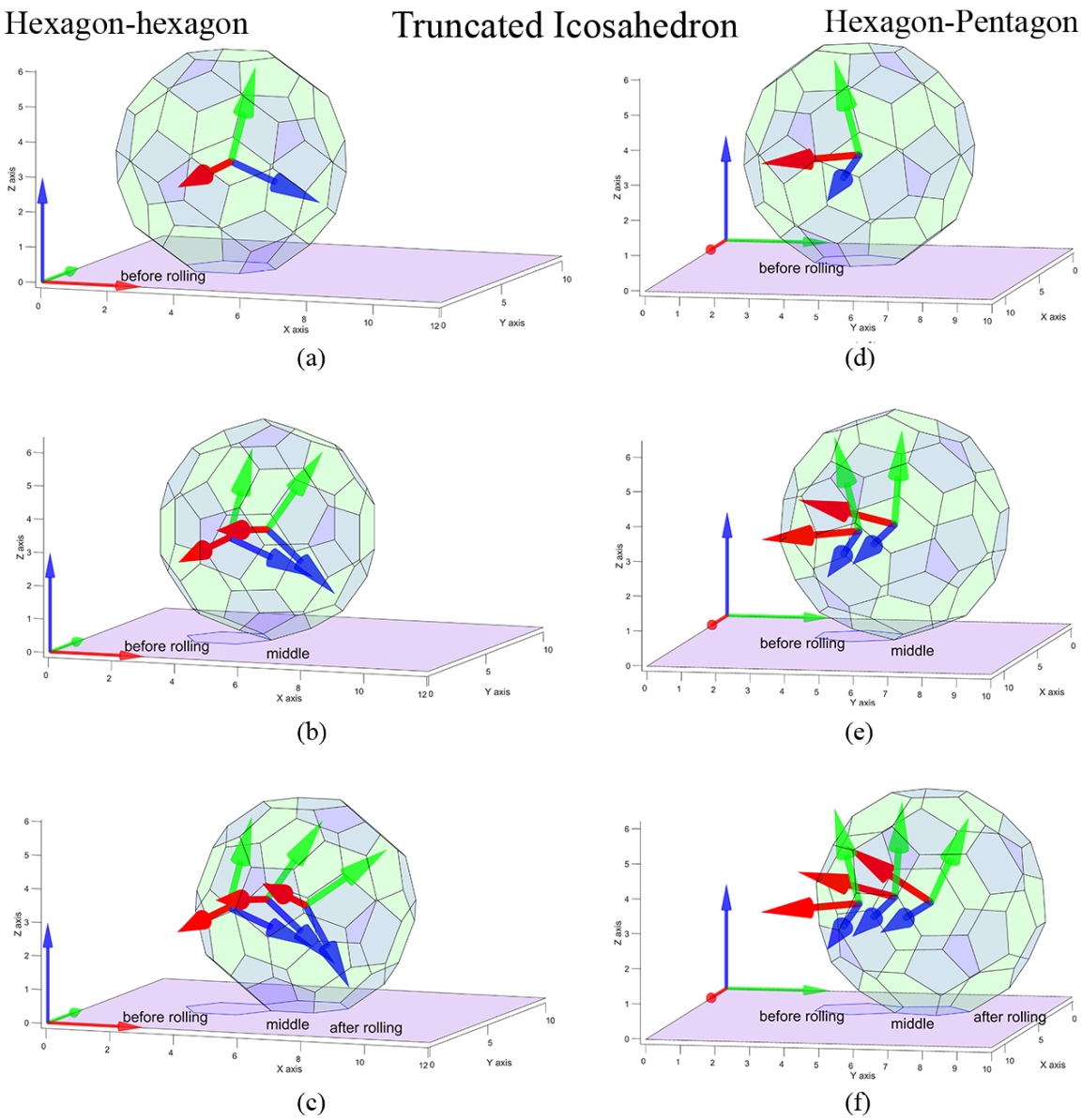


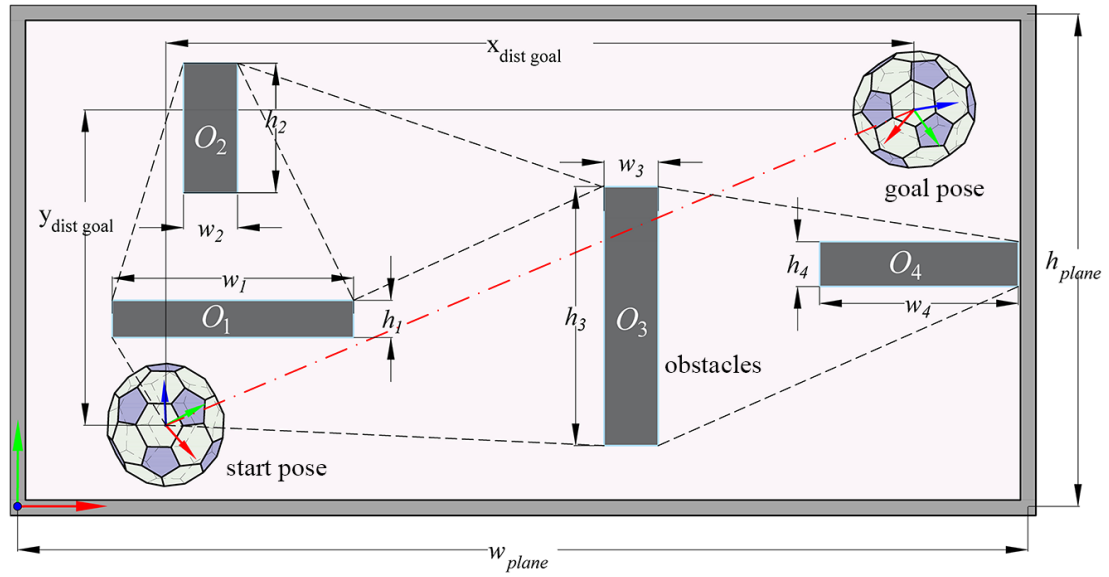
Figure 6.7 The rolling action of a truncated icosahedron from a current surface-based contact to another one. The left column represents a rolling truncated icosahedron from a hexa-based face to a hexa-based face while the Right column shows the rolling truncated icosahedron from a hexa-based face to a hexa-based face. (a)&(d) Starting configureation. (b)&(e) Middle rolling configuration. (c) Finish rolling with $\theta_{66} = \pi - \arccos(-\sqrt{5}/3)$ (f) Finish rolling with $\theta_{65} = \pi - \zeta_{65}$ with the dihedral angle $\zeta_{65} = (\arctan\sqrt{\frac{7+3\sqrt{5}}{2}} + \arctan\left((\tan(\frac{\pi}{5}))\sqrt{\frac{125+41\sqrt{5}}{10}}\right))$

6.4 Results

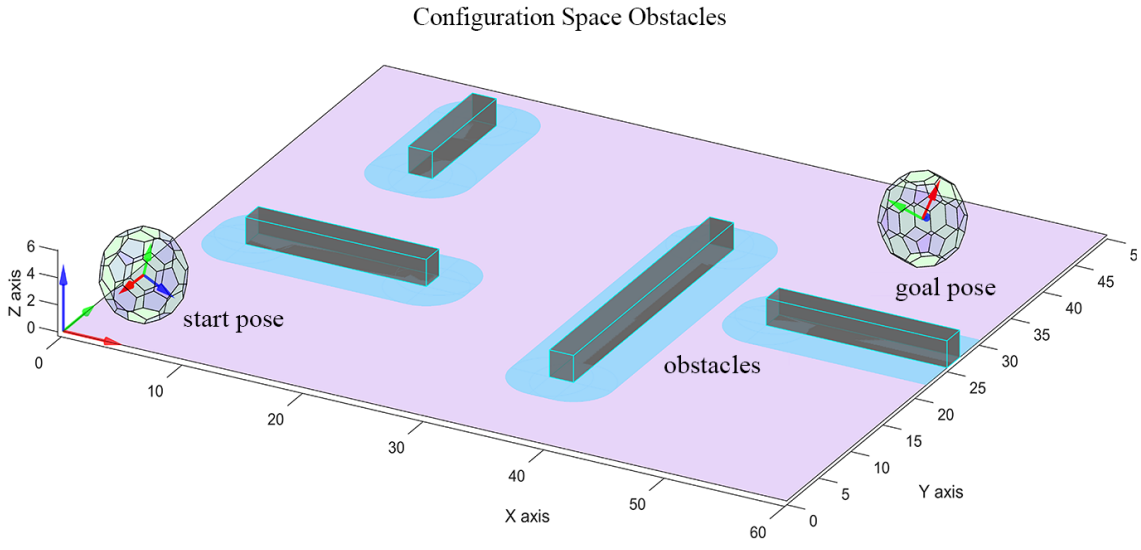
In this section, the path planning RRT-based algorithm in Chapter 4 is applied to find rolling paths of a truncated icosahedron through edge-rolling to reach a goal configuration on a plane. We first describe the environment with obstacles configurations. In the following, the implementation of the path planning algorithm on the environment will be analysed with computational complexity.

6.4.1 Environment Setting

The rolling contact of the truncated icosahedron with obstacle avoidance is considered on a plane and its initial environment is shown in Fig. 6.8. The word coordinate system is represented by $(\mathcal{O}, \mathcal{X}, \mathcal{Y}, \mathcal{Z})$ while a $3D$ cartesian frame $(\mathcal{O}_i, e_1, e_2, e_3)$ is affixed to each center of the initial polyhedron and the goal polyhedron. Given two truncated icosahedrons $\mathcal{P} = (\mathcal{P}_A, \mathcal{P}_B)$ represented by start pose and goal pose, four obstacles $\mathcal{O}_{obs} = (O_1, O_2, O_3, O_4)$, the first goal is to construct the collision free space $\mathcal{W}_{free}$ of the polygon $\mathcal{P}_A$ with respect to $\mathcal{O}_{obs}$, then improved path planning through edge-rolling is executed to achieve the target configuration.



(a)



(b)

Figure 6.8 (a) The environment is set up with four obstacles $\{O_1, O_2, O_3, O_4\}$ and detail dimensions which is represented in the $2D$ map. The red dash-line is the direct distance from the start configuration to the goal configuration. The black dashed lines return the constrained edges in the $\mathcal{O}_{obs}$ (b) The $3D$ view of the $2D$ map with obstacles. The safe zone is calculated for the truncated icosahedron rolling without obstacles collision presented in the light-blue zone.

The environment includes four obstacles with their dimensional as shown in Fig. 6.8. In this study, we only consider static obstacles in a defined plane. The shortest distance from the start pose to the goal pose is the red dashed line which intersects with the obstacles. With the defined obstacle positions on a plane, the non-convex and convex hull can be generated that defines the obstacles potentially reachable. The physical size of obstacles is extended by the radius of the truncated icosahedron sphere, (light blue regions surrounding the obstacles in Fig. 6.8). This forms the collision zone critical for the polyhedron to be able to roll the planned path, without collision through the entire workspace $\mathcal{W}$, until the goal is achieved. During the execution of the path planning algorithm, the step of the best collision-free sample always runs to generate a collision-free contact configuration. The proposed algorithm provides a guaranteed method to find paths that avoid obstacle boundaries as shown in Fig. 6.8.

In the previous section, the truncated icosahedron is denoted as $\mathcal{P}$ and has surface contact with the plane Π. Polyhedron $\mathcal{P}$ consists of 32 surfaces. Giving the index of the 32 surfaces of each truncated icosahedron $\mathcal{P}$ contacted to the plane π determines the configuration of the polygon. For example, Fig. 6.9(a) shows the projections of the position and orientation of the unfolding polyhedron on the plane for the initial configuration at the coordination of $\mathcal{I} = (10, 10, h)$ (h is the distance from the centre of the truncated icosahedron to the plane). The 32 configurations can be determined through this unfolding step. The collision zones are determined based on their fixed geometry coordination systems on the plane. It can be seen from

Fig. 6.9(b) that the obstacle is black, the collision zone is light red and the green zone is the safe area without collision. The red polygon contacts demonstrate direct obstacle collisions while the red-dash polygons-based contacts do not collide with the obstacles directly but lie inside the collision zone. The polyhedra have this type of based-contacts will not consider the expanded nodes in the path planning algorithm. Based on the dimensions of the obstacles and the truncated icosahedron sphere, the CollisionFree function always checks the distance from the centre of the polygon sphere to the boundary of the collision zone. If the distance $d_{collision} \leq 0$, the return value of the algorithm indicates a collision. The other polygon based-contacts indicated by light grey lie in the safe zone which means $d_{collision} > 0$. These expanded points will be checked in the next rolling configuration.

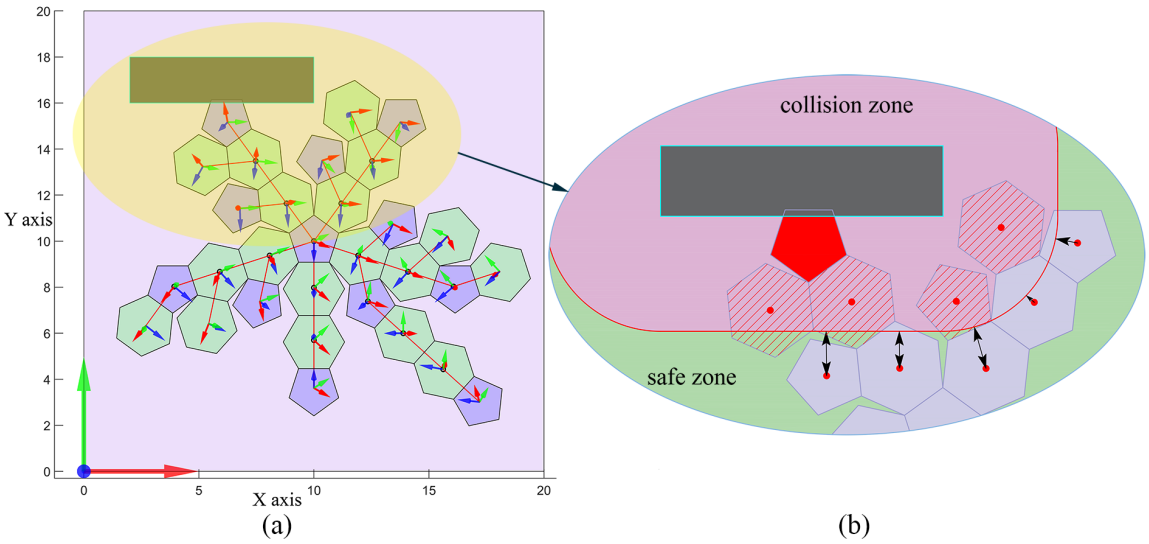


Figure 6.9 Collision checking of the unfolding polygon from path planning RRT-based algorithm. (a) Projections of the orientation of the coordinate system of the unfolding truncated icosahedron on the plane. There are 32 different configurations which will be checked for the obstacle collision. (b) The zoom-in of obstacle collision checking. The red zone indicates the collision area, as discussed in Section 6.4.1 while the green zone is the safe area where the polygon can roll.

6.4.2 Planning for Reaching Goal Configuration

The truncated icosahedron $\mathcal{P}$ is a subspace of the Euclidian space $\mathbb{R}^n$ defined by a set of the hexagon and pentagon faces. Note that this study only considers a truncated icosahedron as a convex polyhedron. The path-finding algorithm-based RRT is run in a rectangle environment with fixed obstacles. All source code of the algorithm was implemented in Matlab and run on a computer with a 3.4 GHz processor and 16GB RAM running the Window 10 operating system. In Fig. 6.12, the tree expansion maintained by the RRT algorithm is shown after 150 iterations of rapidly exploring in the state space. By checking the obstacle regions, the free space is calculated so that reducing the computation time of planning. Moreover, the path planning RRT-based algorithm is improved in its tree node sampling to include paths with smaller costs, which reduces the cost of reaching the goal configuration considerably.

In the simulation, the initial truncated icosahedron contact point coordinates are $\mathcal{P}_{start}^{contact} = [4, 4, 0]^T$ while the starting coordinates of the polyhedron are $\mathcal{P}_{start}^{Poly} = [4, 4, h]^T$, where $h = (3 + \sqrt{5}) * \sqrt{3}/12$ with respect to hexagon face-based contact where the unit length of the edge is 1 (Fig. 6.10).

The goal configuration is located on the top-right of the plane at position $\mathcal{P}_{goal}^{contact} = [50, 40, 0]^T$. The orientation of the goal polygon is random. Fig. 6.11 shows the different positions and orientations of the start and goal configuration. The rotation angle θ_{66} and θ_{65} of the two types of contact-based are the supplementary angle of

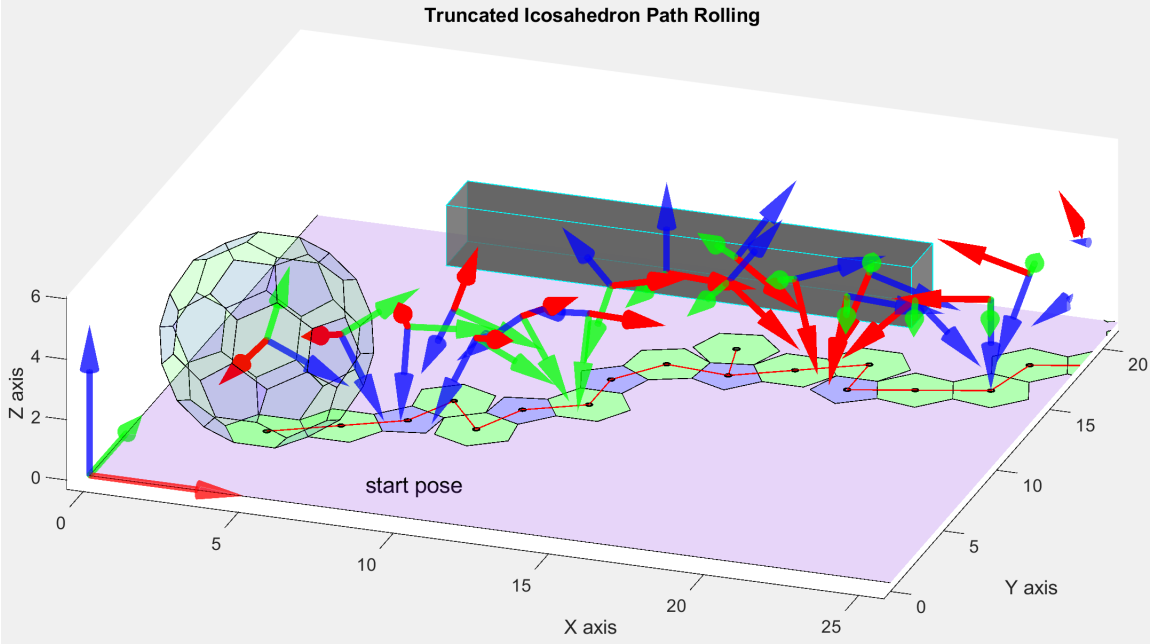


Figure 6.10 The start configuration of the truncated icosahedron belongs to the rolling path with surface-based contacts and its orientations.

their dihedral angles ζ (degree), which is the inner angle between two intersecting faces, as in Table 3.3. The two rolling angles, between the hexagon face and a plane and between the pentagon face and the plane, are represented as $\theta_{66} = \pi - \arccos(-\sqrt{5}/3)$ and $\theta_{65} = \pi - \zeta_{65}$ ($\zeta_{65} = \arctan\sqrt{\frac{7+3\sqrt{5}}{2}} + \arctan\left((\tan(\frac{\pi}{5}))\sqrt{\frac{125+41\sqrt{5}}{10}}\right)$).

In this chapter, the rotation matrix for rolling truncated icosahedron uses the Rodrigues formula [8] represented in Chapter 2 to determine the orientation of the polyhedron from a current position to the next position.

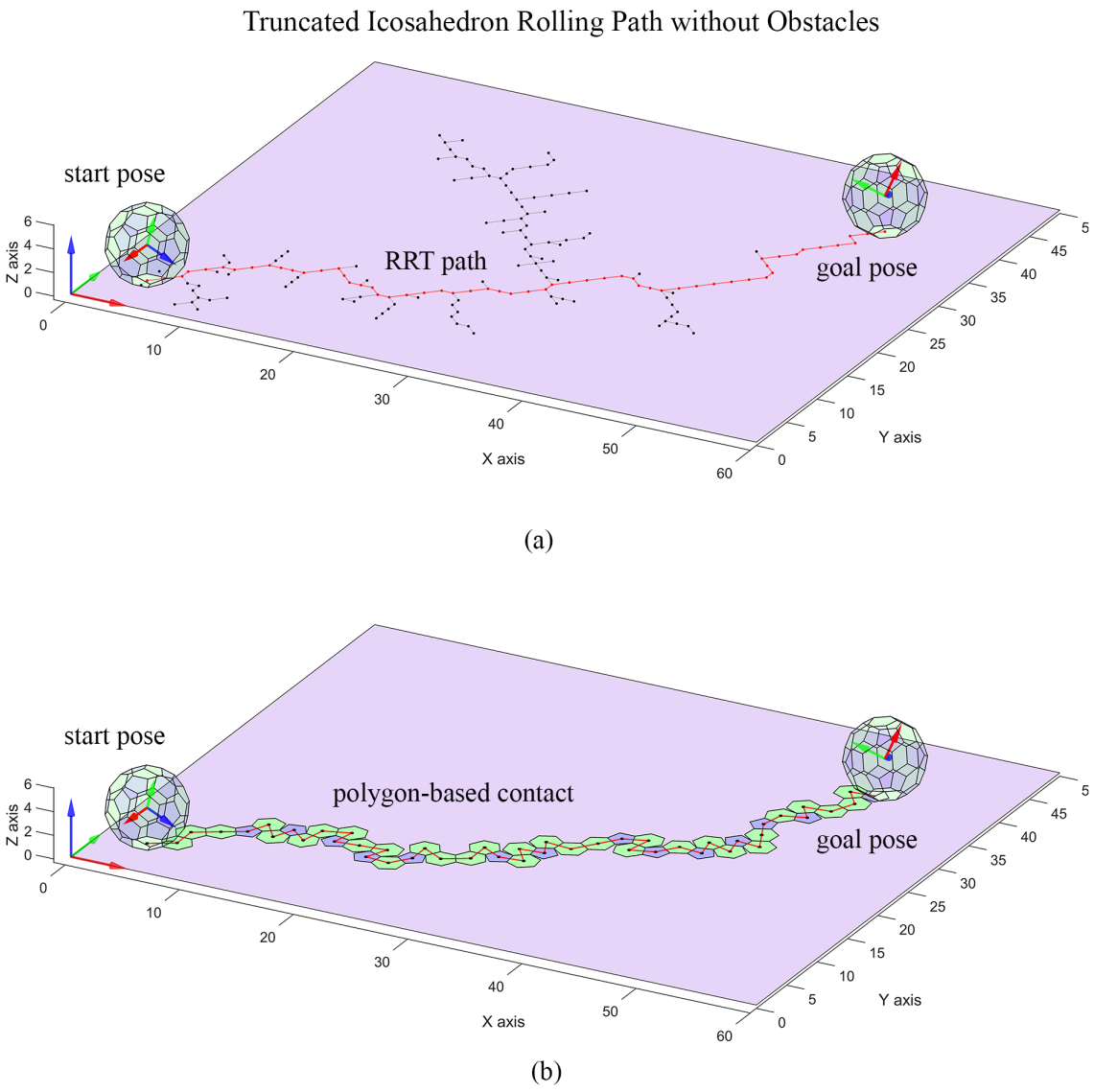


Figure 6.11 The results of path planning based RRT algorithm for the truncated icosahedron through edge-rolling without obstacles on a plane. (a) The $3D$ view of the final RRT path. (b) The polygon contacts belong to the path from a start pose to a goal pose.

The configuration of the rolling truncated icosahedron system is described by the index, position and orientation of the current contact face on the plane and the remaining faces of the polyhedron. Let $q = (F, V, \theta) \in \mathcal{F} \times \mathcal{V} \times \mathbb{R}^3 \times \mathbb{S}^1$ with $\mathcal{F} = \{F_1, ..., F_n\}$ is the set of polyhedron faces and $\mathcal{V} = \{V_1, ..., V_n\}$ is the set of polyhedron vertices. At the beginning of Algorithm 5, all vertices of the truncated

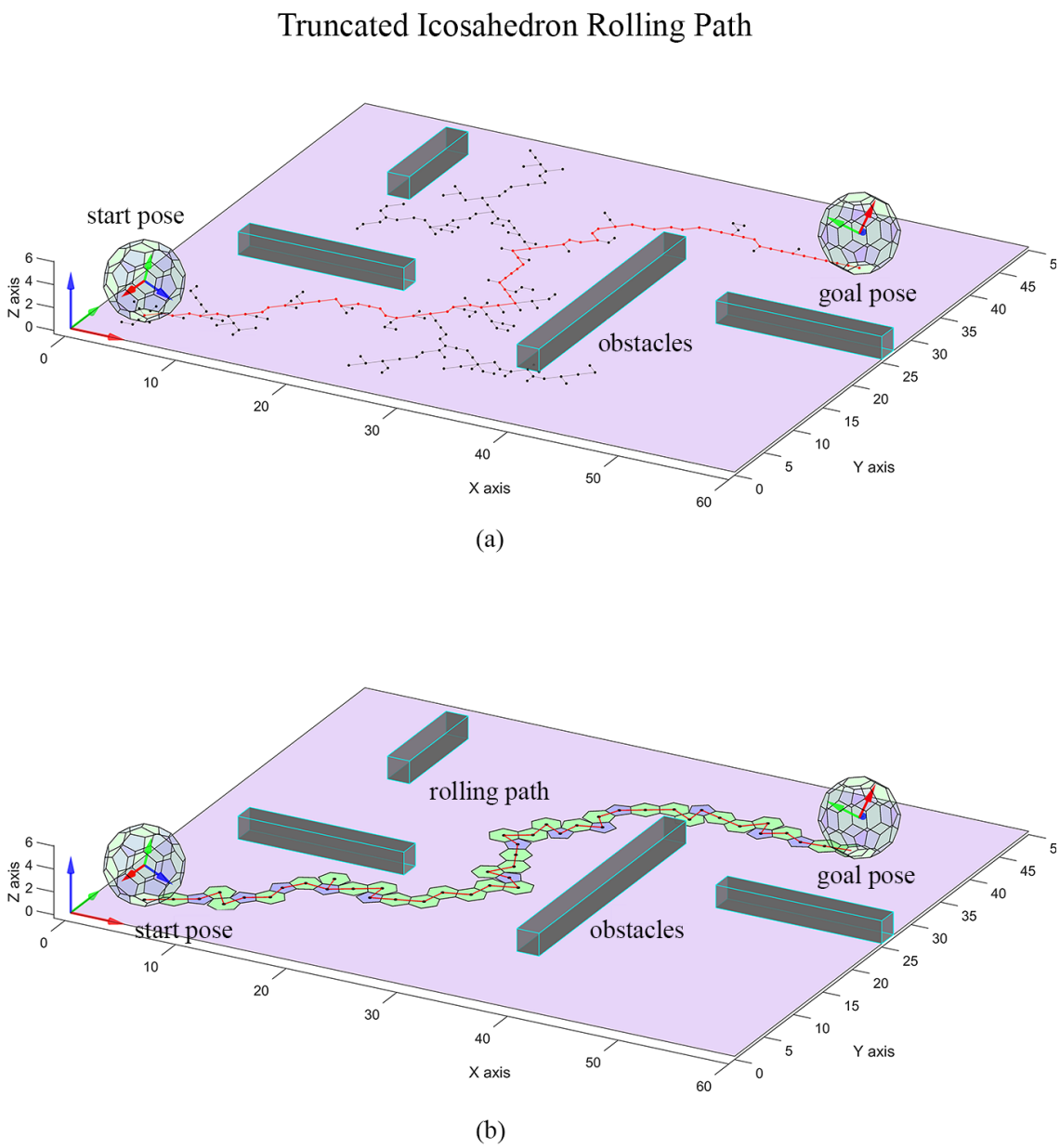


Figure 6.12 The results of path planning based RRT algorithm for the truncated icosahedron through edge-rolling while avoiding obstacles on a plane. (a) The $3D$ view of the final RRT path. (b) The polygon contacts belong to the path from a start pose to a goal pose.

icosahedron are stored in V (Line 2). This data will be stored in $\mathcal{M}_{Poly}$ along with the centre coordinates and orientation. Each of vertex has $3D$ coordinate values $V_i = [x_i, y_i, z_i]$. The polygon-based contacts have $z_i = 0$ which is the condition of maintained contact. At any time of rolling, the surface contact will tranfer to

the line contact which consists of two vertices. The sampling nodes are based on the centre of each surface contact. In the function SAMPLING of the algorithm, the nodes will be transferred to the plane with the $z_{nodes} = 0$ to reduce the calculation of the transformation matrix. The function DISTANCE calculates the distance between node i and node j. The parent configurations are first saved with q_{start}. At early states, there is no obstacle collision, the cost from the start configuration to surrounding nodes is calculated by using the COST function. When a q_{near} is found after selecting q_{rand}, the cost from q_{near} to q_i is compared with other q_{near} cost to find the $q_{nearest}$ nodes. Then the edge between q_{near}^{parent} and q_{near} is eliminated while the new edge between $q_{nearest}$ and q_{new} is generated. Line 7 to Line 11 in Algorithm 5 represent the generated new nodes.

6.4.3 Computational Complexity

The computational complexity of the path planning RRT-based algorithm in the rolling truncated icosahedron depends on the iterations of each function from Algorithm 5 to Algorithm 7. It is noted that the procedures of the functions SAMPLING, STEER, COLLISIONFREE can be performed in a constant number of steps. The NEAREST process is used to find the nearest neighbors which has $O(\log n)$ time in the complexity. The NEAR function executes similar to the nearest neighbour search that can be done in logarithmic time. The function returns the expected number of vertices which can be represented as in $\log n$ time in the linear space. Let $\mathcal{S}_i$ denote

the number of samples of the path planning algorithm in the i iteration, which can be initialised as the random variables. The random variable $O(\log(\mathcal{S}_i)/\mathcal{S}_i)$ is more than a constant in the limit with infinity approaches i [154]. The path planning RRT-based algorithm for rolling with CollisionFree() procedure in the i iteration runs logarithmically $O(\log^n m)$ with the number of nodes expansion in the graph.

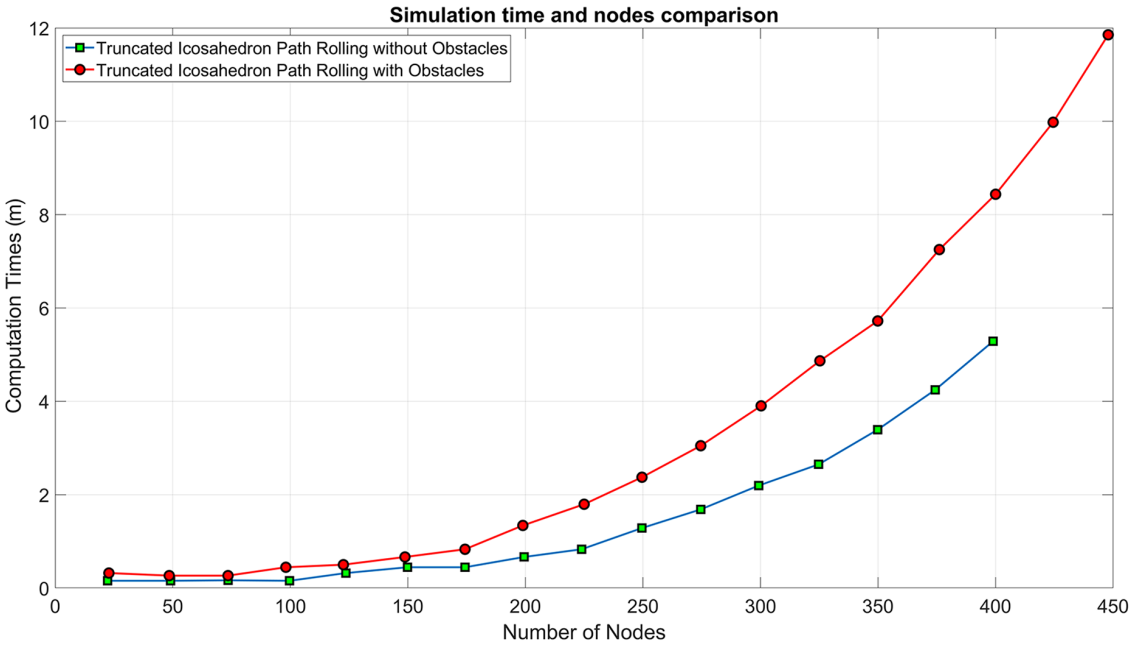


Figure 6.13 The comparison of the time computation and nodes for truncated icosahedron through ege-rolling.

The rolling truncated icosahedron is mainly based on the defined environment with obstacles and the rolling cost between the two states. The tree expansion depends on the defined environment configuration for rolling contact and the goal region with the bias of differential orientation. The cost for the connection between the q_{rand} to the q_{near} is minimized by choosing the nearest node on the tree expansion. We define the cost of edge as $\mathcal{C}_{edge}$ which can be calculated by $\mathcal{C}_{edge} = \mathcal{C} \cdot \text{Distance}(q_i, q_j)$. The sum of all the edge costs is defined by the path cost from the root node q_{init}

to any leaf node q_i. The cost of the new node is $\mathcal{C}_{new} = \mathcal{C}_{edge} + \textsc{Cost}(q_{new}) + \mathcal{C} \cdot \textsc{PolyRoll}(\mathcal{M}_{Poly})$. The two costs to reach the goal configuration included the position and the orientation. The first one is to check the q_{new} whether lying inside the $\mathcal{G}_{goal}$ zone which is decided by ε in which $||q_{new} - q_{goal}|| \leq \varepsilon$. Then the q_{new} orientation will be compared to the goal orientation. If the conditions are met, the node will be saved $q_n \leftarrow q_{new}$. The result of reaching the goal configuration for the truncated icosahedron is shown in Fig. 6.14.

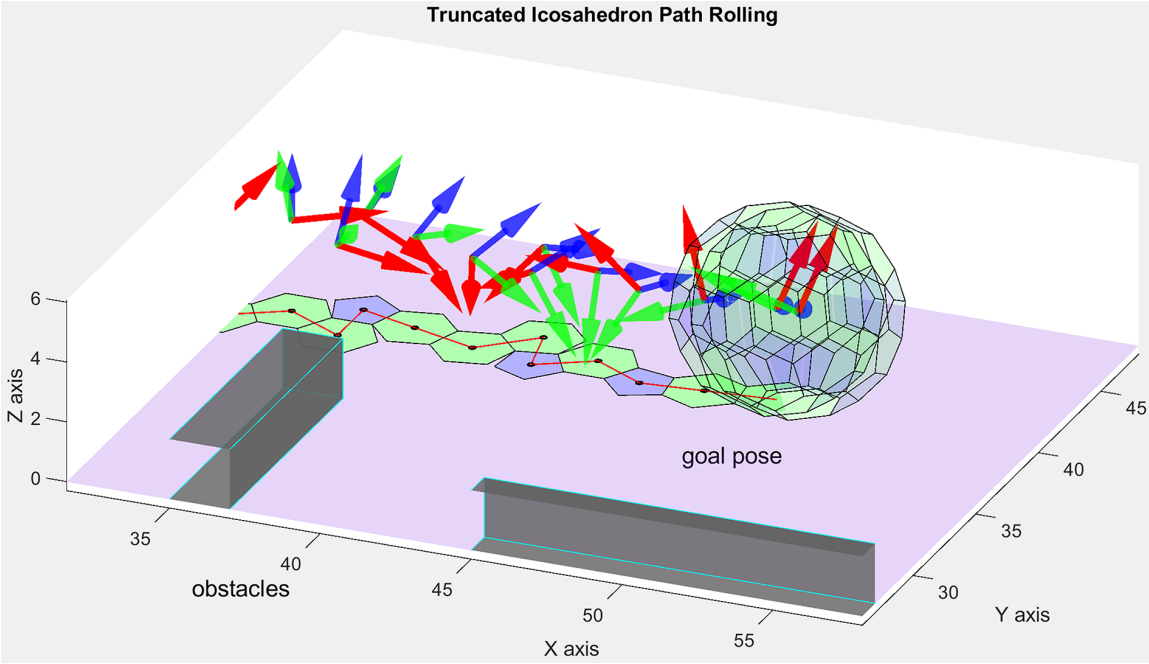


Figure 6.14 The final configuration achieves the goal pose.

6.5 Conclusions and Future work

This chapter presented the results of the improvement of the path planning RRT-based algorithm to the path-finding for the truncated icosahedron through edge-rolling on a plane. We introduced the truncated icosahedron given by truncation of an icosahedron from the Platonic solids family which can be tiled into different flattened geometry shapes, which leave symmetric gaps in the plane. The truncated icosahedron is a special polyhedron that represented generalized soccer-ball patterns. The study then showed successful results in finding the optimal path for rolling the truncated icosahedron from an initial configuration to achieve a random goal configuration by using the path planning RRT-based algorithm. We defined the goal region with the bias of differential orientation that can reduce the executing time to achieve the precise goal configuration.

Chapter 7: Conclusion and Future Work

7.1 General Conclusion of the Thesis

The thesis is dedicated to the application of BFS and RRT path planning methods to solve rolling path tasks for Platonic solids and truncated icosahedron which are represented as convex polyhedra through edge-rolling on a plane. The general conclusions of the thesis are as follows:

Chapter 1 introduced the background, aims and objectives of the thesis.

Chapter 2 showed the background mathematics used in the path planning algorithms in the thesis. The rigid body transformations in the $3D$ space is firstly introduced then the rotation matrix is detailed.

Chapter 3 firstly provided an overview of the polyhedra class, then specified the Platonic solids and truncated icosahedron. The literature on path planning algorithms was presented with their advantages and disadvantages in terms of polyhedra part rolling tasks.

Chapter 4 presented the two path planning methods based on the BFS and RRT algorithms. In the first part, it began with an overview of the BFS algorithm and it then presented the literature for its applications. The most important part is the tree

exploration technique, which is a variation of the BFS was explained. The conditions of node expansions and the complexity in both time and space searching is shown in the next section. In the second part, the discrete path planning based on the RRT algorithm is explained to solve rolling-path finding for the convex polyhedra. An overview of the randomly sampling method and its previous applications are then presented. A potential RRT-based path finding algorithm is analysed to fill the gaps of rolling-polyhedra in the literature and to apply to the truncated icosahedron path-rolling problem.

Chapter 5 mainly showed the significant results of path planning for the Platonic solids through edge-rolling based on the BFS method, which have not been solved in the literature. Related work on rolling polyhedra and previous gaps were introduced belongs to the different discretized grids problems in the path planning. The special rolling case of tetrahedron regarding symmetrical properties is also discussed.

Chapter 6 proposed the path planning method based on RRT technique for the truncated icosahedron through edge-rolling with obstacle avoidance on a plane. The chapter shows significant results of the closest path from the start pose to the goal pose to edge-rolls the truncated icosahedron. This focused on the potential algorithm to solve path planning task for convex polyhedra.

Chapter 7 concluded the thesis, highlighted all of the contributions of the study and suggested a direction for future work.

7.2 Contributions and Main Achievements of the Thesis

The work proposed a comprehensive literature review of path planning methods to solve the problems of path finding for the convex polyhedron on a plane. The work first analysed details of two classes of a convex equilateral polyhedron with polyhedral symmetry, the Platonic solids (the cube, tetrahedron, octahedron, icosahedron, and dodecahedron) and the Archimedean models (the truncated icosahedron with its soccer-ball shape). The potential path planning algorithms are then proposed to find the rolling path for the convex polyhedra on a plane. One inherent problem of discrete search methods is the difficulty of reaching precise targets. This thesis introduced a simple and efficient discrete path planning technique based-BFS and RRT* to address this problem. Each branch of the search is treated as a pose chain of rolling contact. This technique produces motions precisely reaching given targets, and at the same time leads to an earlier successful search termination in several cases with the defined threshold. The contribution of this work is a significant improvement in path planning algorithms to solve more complex rolling-polyhedra tasks.

The work adopted a path planning method based on the BFS algorithm for the Platonic solids through edge-rolling from an initial pose to a goal pose. Each type of the Platonic solids needs to roll on its own prescribed grid. The path planning

method is divided into two approaches to finding the rolling path - the direct path from an initial pose to the goal pose and the closest path at the goal position to find the goal configuration. Path planning based on BFS can find multiple solutions and has a low computation time for searching the rolling path to reach the goal configuration in both limited and large search spaces. This contribution will lead to the improved discovery of path planning for rolling convex polyhedra on a plane.

The work further proposed a discrete path planning method -the RRT-based algorithm for the truncated icosahedron-high complex convex polyhedron, through edge-rolling to achieve a random goal configuration on a plane. It consisted of checking orientation for each step through unfolding polyhedron and checking safety distance to obstacles to generate the optimal path. The work shows the existing path for achieving the goal pose and the closed path in the case of non-achieving goal orientation due to symmetric and polygon tiling properties. The contribution will lead to the improved discovery of potential paths for rolling regular convex polyhedra on a convex surfaces.

Additionally, the work analysed the rolling constraints based on the convex polyhedra that a polyhedron can be brought to reach by rolling on a plane about its edges. The problem is important to practical applications, such as automatic part manipulation, as well as being theoretically stimulating. As a result of the analysis, the work pointed out that proposed path planning methods may show a potential successful path-finding algorithms for different polyhedra on the different regular surfaces, which were

analysed previously. Indeed this work not only developed path planning algorithms but solidly validated them in the simulation.

7.3 Future Work

The thesis focused on developing path planning algorithms based on BFS and RRT techniques to find the optimal path for the Platonic solids and truncated icosahedron from an initial pose to a goal pose. It then validated the algorithms in simulation for each of the polyhedra. Future research should be directed towards the following aspects:

- Optimising the path planning algorithms for general convex polyhedra which roll in deformable surfaces. The algorithm will be modified to reduce the computational complexity of for searching the nearest neighbour and achieving a more precise target configuration.
- Validating the path planning methods on more convex polyhedra on a convex surface.
- Considering improvements of the algorithms to robotic in-hand manipulation applications.

List of Publications Arisen from this PhD Study

Journal paper

Lam NT, Howard I, Cui L (2021) "Path planning for the Platonic solids on prescribed grids by edge-rolling". *PLoS ONE* 16(6): e0252613. https://doi.org/10.1371/journal.pone.0252613.

Lam NT, Howard I, Cui L (2022) "Discrete Path Planning Based RRT Algorithm for Truncated Icosahedron through Edge-Rolling on a Plane". *Journal of IEEE Intelligent Systems* (to be submitted).

Conference

N. T. Lam, I. Howard and L. Cui, "A Review of Trajectory Planning for Autonomous Excavator in Mining and Construction Sites",*10th Australasian Congress on Applied Mechanics(ACAM)*, 2021. doi:10.3316/informit.323406814564895.

N. T. Lam, I. Howard and L. Cui, "A Literature Review on Path Planning of Polyhedrons with Rolling Contact", *2019 4th International Conference on Control, Robotics and Cybernetics (CRC)*, 2019, pp. 145-151, doi:10.1109/CRC.2019.00038.

Bibliography

[1] S. M. LaValle. *Planning Algorithms*. 1st. Cambridge: Cambridge University Press, 2006, p. 842. URL: http://msl.cs.uiuc.edu/planning/ (cit. on pp. 1, 15, 80).

[2] A. Bry and N. Roy. "Rapidly-exploring Random Belief Trees for motion planning under uncertainty". In: *Proceedings - IEEE International Conference on Robotics and Automation*. May 2011, pp. 723–730. DOI: 10.1109/ICRA.2011.5980508 (cit. on p. 1).

[3] E. Plaku and G. D. Hager. "Sampling-Based Motion and Symbolic Action Planning with geometric and differential constraints". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 5002–5008. DOI: 10.1109/ROBOT.2010.5509563 (cit. on p. 1).

[4] U. Alphan et al. "Robust Multi-Robot Optimal Path Planning with Temporal Logic Constraints". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 4693–4698. DOI: 10.1109/ICRA.2012.6224792 (cit. on p. 1).

[5] R. Surovec et al. "A conceptual design of the self-reconfigurable mobile robot Wheeking 1". In: *ATP Journal Plus* 1 (Jan. 2011), pp. 57–60 (cit. on p. 1).

[6] K. Buchin et al. "On Rolling Cube Puzzles". In: *CCCG 2007 - 19th Canadian Conference on Computational Geometry*. Jan. 2007, pp. 141–144 (cit. on pp. 2, 50).

[7] R. M. Murray, S. S. Sastry, and Z. Li. *A Mathematical Introduction to Robotic Manipulation*. 1st. USA: CRC Press, Inc., 1994. ISBN: 0849379814 (cit. on pp. 8, 30).

[8] J. S. Dai. "Euler-Rodrigues formula variations, quaternion conjugation and intrinsic connections". In: *Mechanism and Machine Theory* 92 (2015), pp. 144–152 (cit. on pp. 11, 120).

[9] J. S. Dai. *Geometrical foundations and screw algebra for mechanisms and robotics*. 2014 (cit. on p. 13).

[10] H. Zhang, W. Lin, and A. Chen. "Path Planning for the Mobile Robot: A Review". In: *Symmetry* 10.10 (2018), p. 450. ISSN: 2073-8994. DOI: 10.3390/sym10100450. URL: https://www.mdpi.com/2073-8994/10/10/450 (cit. on pp. 15, 35, 36).

[11] L. E. Kavraki and J.C. Latombe. “Probabilistic Roadmaps for Robot Path Planning”. In: *Practical Motion Planning in Robotics: Current Approaches and Future Directions* (1998) (cit. on pp. 15, 43).

[12] C. S. Cai and B. Roth. “On the planar motion of rigid bodies with point contact”. In: *Mechanism and Machine Theory* 21.6 (1986), pp. 453–466. ISSN: 0094-114X. DOI: 10.1016/0094-114X(86)90128-X. URL: http://www.sciencedirect.com/science/article/pii/0094114X8690128X (cit. on pp. 15, 32).

[13] C. S. Cai and B. Roth. “On the spatial motion of a rigid body with point contact”. In: *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*. Vol. 4, pp. 686–695. DOI: 10.1109/ROBOT.1987.1087971 (cit. on pp. 15, 34).

[14] C. S. Cai and B. Roth. “On the spatial motion of a rigid body with line contact”. In: *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*. Vol. 2, pp. 1036–1041. DOI: 10.1109/ROBOT.1988.12197 (cit. on p. 15).

[15] A. V. Borisov, Y. N. Fedorov, and I. S. Mamaev. “Chaplygin ball over a fixed sphere: an explicit integration”. In: *Regular and Chaotic Dynamics* 13.6 (2008), pp. 557–571. ISSN: 1468-4845. DOI: 10.1134/S1560354708060063. URL: https://doi.org/10.1134/S1560354708060063 (cit. on pp. 15, 30).

[16] A. Bicchi, Y. Chitour, and A. Marigo. “Reachability and steering of rolling polyhedra: a case study in discrete nonholonomy”. In: *IEEE Transactions on Automatic Control* 49.5 (2004), pp. 710–726 (cit. on p. 15).

[17] Y.Aiyama, M. Inaba, and H. Inoue. “Pivoting: A new method of graspless manipulation of object by robot fingers”. In: *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. 1993, pp. 136–143. DOI: 10.1109/IROS.1993.583091 (cit. on pp. 16, 100).

[18] M. Erdmann, M. T. Mason, and G. Jr. Vaněček. “Mechanical parts orienting: the case of a polyhedron on a table”. In: *Proceedings of 1991 IEEE International Conference on Intelligent Robots and Systems*. Vol. 1. 1993, pp. 360–365. DOI: 10.1109/ROBOT.1991.13160 (cit. on pp. 16, 34, 50, 80, 100, 104).

[19] J. E. Goodman, J. O’Rourke, and C. D. Toth. *Handbook of Discrete and Computational Geometry*. 3rd. New York: Chapman and Hall/CRC, 2017, p. 1948. ISBN: 9781315119601. DOI: https://doi.org/10.1201/9781315119601 (cit. on p. 17).

[20] F. S. Fadzli et al. “Robotic Indoor Path Planning Using Dijkstra’s Algorithm with Multi-Layer Dictionaries”. In: *2015 2nd International Conference on Information Science and Security (ICISS)*. 2015, pp. 1–4. DOI: 10.1109/ICISSEC.2015.7371031 (cit. on pp. 17, 39).

[21] R. A. Brooks and T. Lozano-Perez. “A Subdivision Algorithm in Configuration Space for Findpath with Rotation”. In: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*. Karlsruhe, West Germany: Morgan Kaufmann Publishers Inc., 1983, pp. 799–806 (cit. on p. 17).

[22] A. Bicchi, D. Prattichizzo, and S. S. Sastry. “Planning motions of rolling surfaces”. In: *Proceedings of 1995 34th IEEE Conference on Decision and Control*. Vol. 3. 1995, pp. 2812–2817. ISBN: 0191-2216. DOI: 10.1109/CDC.1995.478544 (cit. on pp. 17, 100).

[23] M. Lin and S. Gottschalk. “Collision Detection Between Geometric Models: A Survey”. In: *IMA Conf. on Mathematics of Surfaces* 8 (June 1998) (cit. on p. 18).

[24] R. Haghnazar, H Nooshin, and M. Golabchi. “Improving the Regularity of Geodesic Domes Using the Concept of Stepping Projection”. In: *International Journal of Space Structures* 29 (June 2014), pp. 81–96. DOI: 10.1260/0266-3511.29.2.81 (cit. on p. 19).

[25] M. Kline. *Mathematics in Western Culture*. London: Oxford University Press, 2008 (cit. on p. 20).

[26] R. D. Archer-Hind. *The Timaeus of Plato*. London: McMillan and Co., 1888 (cit. on p. 20).

[27] J. Kepler. *Mysterium Cosmographicum*. New York: Abaris Books, 1981 (cit. on p. 21).

[28] P. F. Damasceno, M. Engel, and S. C. Glotzer. “Predictive Self-Assembly of Polyhedra into Complex Structures”. In: *Science* 337.6093 (2012), pp. 453–457. DOI: 10.1126/science.1220869 (cit. on p. 21).

[29] Y. Geng et al. “Engineering entropy for the inverse design of colloidal crystals from hard shapes”. In: *Science* 5.7 (2019). DOI: 10.1126/sciadv.aaw0514 (cit. on p. 21).

[30] B. Cipra. “How to Play Platonic Billiards”. In: *Science* 275.5303 (1997), p. 1070. DOI: 10.1126/science.275.5303.1070 (cit. on p. 21).

[31] C. Leopold. *Geometric and Aesthetic Concepts Based on Pentagonal Structures*. Sriraman B., Ed. Springer International Publishing, 2019 (cit. on pp. 21, 83).

[32] J. Kepler et al. *Prodromus dissertationum cosmographicarum, continens mysterium cosmographicum, de admirabili proportione orbium coelestium, deque causis coelorum numeri, magnitudinis, motuumque periodicorum genuinis and propriis, demonstratum, per quinque regularia corpora geometrica*. Tubingae: excudebat Georgius Gruppenbachius, 1596. DOI: 10.3931/e-rara-445. URL: https://doi.org/10.3931/e-rara-445 (cit. on p. 22).

[33] P. R. Cromwell. *Polyhedra*. New York: Cambridge University Press, 1999 (cit. on pp. 23, 105).

[34] M. Livio. *The Golden Ratio: The Story of Phi, the World's Most Astonishing Number*. New York City: Broadway Books, 2002. ISBN: 78-0-7679-0816-0 (cit. on p. 28).

[35] J. H. Beynon, J. E. Garnham, and K. J. Sawley. "Rolling contact fatigue of three pearlitic rail steels". In: *Wear* 192.1 (1996), pp. 94–111. ISSN: 0043-1648. DOI: 10.1016/0043-1648(95)06776-0. URL: http://www.sciencedirect.com/science/article/pii/0043164895067760 (cit. on p. 30).

[36] G. E. Yakubov et al. "Tribology of particle suspensions in rolling-sliding soft contacts". In: *Biotribology* 3 (2015), pp. 1–10. ISSN: 2352-5738. DOI: 10.1016/j.biotri.2015.09.003. URL: http://www.sciencedirect.com/science/article/pii/S2352573815300056 (cit. on p. 30).

[37] David J. Montana. "The Kinematics of Contact and Grasp". In: *The International Journal of Robotics Research* 7.3 (1988), pp. 17–32. ISSN: 0278-3649. DOI: 10.1177/027836498800700302 (cit. on pp. 30, 31, 34).

[38] A. Marigo and A.. Bicchi. "Rolling bodies with regular surface: controllability theory and applications". In: *IEEE Transactions on Automatic Control* 45.9 (2000), pp. 1586–1599. ISSN: 0018-9286. DOI: 10.1109/9.880610 (cit. on pp. 30, 34, 80, 103).

[39] Z. Li and J. F. Canny. "Nonholonomic Motion Planning". In: *1991 IEEE International Conference on Robotics and Automation* 192 (1991), pp. XV, 448. ISSN: 0893-3405. DOI: 10.1007/978-1-4615-3176-0 (cit. on p. 30).

[40] H. Maekawa, K. Tanie, and K. Komoriya. "Kinematics, statics and stiffness effect of 3D grasp by multifingered hand with rolling contact at the fingertip". In:

Proceedings of International Conference on Robotics and Automation. Vol. 1. 1997, pp. 78–85. DOI: 10.1109/ROBOT.1997.620019 (cit. on p. 30).

[41] R. Muszyński and K. Tchoń. "Singularities and Mobility of Nonholonomic Systems: The Ball Rolling on a Plane". In: *IFAC Proceedings Volumes* 33.27 (2000), pp. 593–598. ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)37995-8 (cit. on p. 30).

[42] E. Hartmann. "G/sup n/-blending with rolling ball contact curves". In: *Proceedings Geometric Modeling and Processing 2000. Theory and Applications*. 2000, pp. 385–389. DOI: 10.1109/GMAP.2000.838269 (cit. on p. 31).

[43] R. W. Brockett and Liyi Dai. "Non-holonomic Kinematics and the Role of Elliptic Functions in Constructive Controllability". In: *Nonholonomic Motion Planning*. Ed. by Zexiang Li and J. F. Canny. Boston, MA: Springer US, 1993, pp. 1–21. ISBN: 978-1-4615-3176-0. DOI: 10.1007/978-1-4615-3176-0_1 (cit. on p. 31).

[44] N. Sarkar, Yun Xiaoping, and V. Kumar. "Dynamic control of 3-D rolling contacts in two-arm manipulation". In: *IEEE Transactions on Robotics and Automation* 13.3 (1997), pp. 364–376. ISSN: 1042-296X. DOI: 10.1109/70.585899 (cit. on p. 32).

[45] S. Arimoto, M. Yoshida, and J. Bae. "Dynamic force/torque closure for 2D and 3D objects by means of rolling contacts with robot fingers". In: *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*. Vol. 1. 2003, pp. 178–183. DOI: 10.1109/RISSP.2003.1285570 (cit. on p. 32).

[46] X. Yun et al. "Control of multiple arms with rolling constraints". In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. Vol. 3. 1992, pp. 2193–2198. DOI: 10.1109/ROBOT.1992.219932 (cit. on p. 32).

[47] M. Zribi, J. Chen, and M. S. Mahmoud. "Coordination and Control of Multifingered Robot Hands with Rolling and Sliding Contacts". In: *Journal of Intelligent and Robotic Systems* 24.2 (1999), pp. 125–149. ISSN: 1573-0409. DOI: 10.1023/A:1008097528538. URL: https://link.springer.com/content/pdf/10.1023/A:1008097528538.pdf (cit. on p. 32).

[48] B. Kiss, J. Lévine, and B. Lantos. "On Motion Planning for Robotic Manipulation with Permanent Rolling Contacts". In: *The International Journal of Robotics Research* 21.5-6 (2002), pp. 443–461. ISSN: 0278-3649. DOI: 10.1177/027836402321261959 (cit. on p. 32).

[49] M. Svinin and S. Hosoe. “Motion Planning Algorithms for a Rolling Sphere With Limited Contact Area”. In: *IEEE Transactions on Robotics* 24.3 (2008), pp. 612–625. ISSN: 1552-3098. DOI: 10.1109/TRO.2008.921568 (cit. on p. 32).

[50] A. M. Okamura, N. Smaby, and M. R. Cutkosky. “An overview of dexterous manipulation”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 1. 2000, pp. 255–262. ISBN: 1050-4729. DOI: 10.1109/ROBOT.2000.844067 (cit. on p. 32).

[51] L. Cui and J. S. Dai. “Geometric Kinematics of Rigid Bodies with Point Contact”. In: *Advances in Robot Kinematics: Motion in Man and Machine*. Ed. by Jadran Lenarcic and Michael M. Stanisic. Springer Netherlands, 2010, pp. 429–436. ISBN: 978-90-481-9262-5 (cit. on p. 32).

[52] L. Cui and J. S. Dai. “A Polynomial Approach to Inverse Kinematics of Rolling Contact”. In: *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* 4.45035 (2012). 10.1115/DETC2012-70852, pp. 1563–1570. DOI: 10.1115/DETC2012-70852 (cit. on p. 32).

[53] L. Cui and J. S. Dai. “A Polynomial Formulation of Inverse Kinematics of Rolling Contact”. In: *Journal of Mechanisms and Robotics* 7.4 (2015). 10.1115/1.4029498, pp. 041003–041003–9. ISSN: 1942-4302. DOI: 10.1115/1.4029498. URL: http://dx.doi.org/10.1115/1.4029498 (cit. on p. 32).

[54] L. Cui and J. S. Dai. “Rolling Contact in Kinematics of Multifingered Robotic Hands”. In: *Advances in Robot Kinematics 2016*. Ed. by Jadran Lenarčič and Jean-Pierre Merlet. Cham: Springer International Publishing, 2018, pp. 217–224. ISBN: 978-3-319-56802-7. DOI: 10.1007/978-3-319-56802-7_23. URL: https://doi.org/10.1007/978-3-319-56802-7_23 (cit. on p. 32).

[55] H. Cartan. *Differential Forms*. New York: Dover Publisher, 1996 (cit. on p. 32).

[56] É. J. Cartan. *Riemannian Geometry in an Orthogonal Frame*. Singapore: World Scientic, 2002 (cit. on p. 32).

[57] E. L. Mansfield. “Algorithms for Symmetric Differential Systems”. In: *Foundations of Computational Mathematics* 1.4 (2001), pp. 335–383. DOI: 10.1007/s002080010014 (cit. on p. 33).

[58] M. Oleg. “Moving coframes and symmetries of differential equations”. In: *Journal of Physics A: Mathematical and General* 35.12 (2002), p. 2965 (cit. on p. 33).

[59] G. M. Beffa. “Poisson Geometry of Differential Invariants of Curves in Some Nonsemisimple Homogeneous Spaces”. In: *Proceedings of the American Mathematical Society*. Vol. 134. 3. 2006, pp. 779–791 (cit. on p. 33).

[60] J. Patera V. Boyko and R. Popovych. “Computation of invariants of Lie algebras by means of moving frames”. In: *Journal of Physics A: Mathematical and General* 39.20 (2006), p. 5749 (cit. on p. 33).

[61] V. Boyko, J. Patera, and R. Popovych. “Invariants of solvable lie algebras with triangular nilradicals and diagonal nilindependent elements”. In: *Linear Algebra and its Applications* 428.4 (2008), pp. 834–854. DOI: 10.1016/j.laa.2007.08.017 (cit. on p. 33).

[62] L. Cui and J. S. Dai. “A Darboux-Frame-Based Formulation of Spin-Rolling Motion of Rigid Objects With Point Contact”. In: *IEEE Transactions on Robotics* 26.2 (2010), pp. 383–388. DOI: 10.1109/TRO.2010.2040201 (cit. on p. 33).

[63] O. Faugeras. “Cartan’s moving frame method and its application to the geometry and evolution of curves in the euclidean, affine and projective planes”. In: *Applications of Invariance in Computer Vision, Berlin, Heidelberg* (1994), pp. 9–46. DOI: 10.1007/3-540-58240-1_2 (cit. on p. 33).

[64] A. Bicchi, A. Marigo, and D. Prattichizzo. “Dexterity through rolling: manipulation of unknown objects”. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*. Vol. 2. 1999, pp. 1583–1588. ISBN: 1050-4729. DOI: 10.1109/ROBOT.1999.772585 (cit. on p. 33).

[65] A. Marigo, Y. Chitour, and A. Bicchi. “Manipulation of Polyhedral parts by rolling”. In: *Proc. IEEE Int. Conf on Robotics and Automation*. Vol. 4. 1997, pp. 2992–2997. DOI: 10.1109/ROBOT.1997.606742 (cit. on p. 33).

[66] A. A. Cole, P. Hsu, and S. S. Sastry. “Dynamic control of sliding by robot hands for regrasping”. In: *IEEE Transactions on Robotics and Automation* 8.1 (1992), pp. 42–52. ISSN: 1042-296X. DOI: 10.1109/70.127238 (cit. on p. 34).

[67] Z. Li and J. Canny. “Motion of two rigid bodies with rolling constraint”. In: *IEEE Transactions on Robotics and Automation* 6.1 (1990), pp. 62–72. ISSN: 1042-296X. DOI: 10.1109/70.88118 (cit. on pp. 34, 101, 103).

[68] A. Marigo et al. “Planning Motions of Polyhedral Parts by Rolling”. In: *Algorithmica* 26.3 (2000), pp. 560–576. ISSN: 1432-0541. DOI: 10.1007/s004539910024 (cit. on pp. 34, 50, 100, 104).

[69] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. "Hamilton Paths in Grid Graphs". In: *SIAM Journal on Computing* 11.4 (1982), pp. 676–686. DOI: 10.1137/0211056 (cit. on pp. 34, 81, 104).

[70] L. Yang et al. "Survey of Robot 3D Path Planning Algorithms". In: *Journal of Control Science and Engineering* 2016 (2016), p. 22. DOI: 10.1155/2016/7426913 (cit. on p. 35).

[71] J. Tisdale, Z. Kim, and J. K. Hedrick. "Autonomous UAV path planning and estimation". In: *IEEE Robotics & Automation Magazine* 16.2 (2009), pp. 35–42. ISSN: 1070-9932. DOI: 10.1109/MRA.2009.932529 (cit. on p. 36).

[72] U. Lee et al. "Local path planning in a complex environment for self-driving car". In: *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*. 2014, pp. 445–450. DOI: 10.1109/CYBER.2014.6917505 (cit. on p. 36).

[73] J. Thomas, A. Blair, and N. Barnes. "Towards an efficient optimal trajectory planner for multiple mobile robots". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. Vol. 3. 2003, pp. 2291–2296. DOI: 10.1109/IROS.2003.1249212 (cit. on pp. 36, 80).

[74] F. Lamiraux and J. Lammond. "Smooth motion planning for car-like vehicles". In: *IEEE Transactions on Robotics and Automation* 17.4 (2001), pp. 498–501. DOI: 10.1109/70.954762 (cit. on pp. 37, 80).

[75] P. Cheng et al. "RRT-Based Trajectory Design for Autonomous Automobiles and Spacecraft". In: *Archives of Control Sciences* 11.3-4 (2001), pp. 167–194. DOI: citeulike-articleid:4232681 (cit. on p. 37).

[76] D. C. Conner, A. A. Rizzi, and H. Choset. "Composition of local potential functions for global robot control and navigation". In: *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. Vol. 4. 2003, pp. 3546–3551. DOI: 10.1109/IROS.2003.1249705 (cit. on p. 37).

[77] E.W. Dijkstra. "A Note on Two Problems in Connexion with Graphs." In: *Numerische Mathematik* 1 (1959), pp. 269–271. URL: http://eudml.org/doc/131436 (cit. on pp. 37, 38).

[78] Y. K. Hwang and N. Ahuja. "A potential field approach to path planning". In: *IEEE Transactions on Robotics and Automation* 8.1 (1992), pp. 23–32. ISSN: 1042-296X. DOI: 10.1109/70.127236 (cit. on p. 38).

[79] P. E. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968). DOI: 10.1109/TSSC.1968.300136 (cit. on pp. 39, 61).

[80] A. Stentz. "Optimal and efficient path planning for partially-known environments". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. Vol. 4. 1994, pp. 3310–3317. DOI: 10.1109/ROBOT.1994.351061 (cit. on p. 40).

[81] A. Nash et al. *Theta*: any-angle path planning on grids*. Conference paper. 2007 (cit. on p. 40).

[82] A. Nash, S. Koenig, and C. Tovey. "Lazy Theta*: Any-angle path planning and path length analysis in 3D". In: *Twenty-Fourth AAAI Conference on Artificial Intelligence*. 2010 (cit. on p. 40).

[83] C. Berge. *The Theory of Graphs and Its Applications*. Metheum Co. Ltd., London, England, 1962 (cit. on p. 41).

[84] R. E. Bellman and Stuart E. D. *Applied Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1962 (cit. on pp. 41, 54).

[85] R. Busacker and T. Saaty. *Finite Graphs and Networks*. McGraw Hill Company, New York, 1965 (cit. on p. 41).

[86] A. Kaufmann. *Graphs, Dynamic Programming and Finite Games*. Academic Press, New York, 1967 (cit. on p. 41).

[87] J. Doran and D. Michie. "Experiments with the graph traverser program". In: *Proceedings of the Royal Society (A)*. Vol. 294. 1966, pp. 235–259. DOI: http://doi.org/10.1098/rspa.1966.0205 (cit. on p. 41).

[88] M. M. Costa and M. F. Silva. "A Survey on Path Planning Algorithms for Mobile Robots". In: *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2019, pp. 1–7. DOI: 10.1109/ICARSC.2019.8733623 (cit. on p. 41).

[89] F. F. Li and K. J. Jia. "Path planning and smoothing of mobile robot based on improved artificial fish swarm algorithm". In: *Scientific Reports* 12 (1 2022), pp. 659–665. ISSN: 2045-2322. DOI: 10.1038/s41598-021-04506-y (cit. on p. 41).

[90] L. Larsen et al. “Automatic Path Planning of Industrial Robots Comparing Sampling-based and Computational Intelligence Methods”. In: *Procedia Manufacturing* 11 (2017), pp. 2351–9789. ISSN: 2045-2322. DOI: 10.1016/j.promfg.2017.07.237. URL: https://www.sciencedirect.com/science/article/pii/S2351978917304456 (cit. on p. 41).

[91] Y. Y. Ahn et al. “Analysis of Topological Characteristics of Huge Online Social Networking Services”. In: *WWW '07: Proceedings of the 16th international conference on World Wide Web*. New York, NY, USA: Association for Computing Machinery, 2007, pp. 835–844. DOI: 10.1145/1242572.1242685. URL: https://doi.org/10.1145/1242572.1242685 (cit. on p. 42).

[92] A. Mislove et al. “Measurement and Analysis of Online Social Networks”. In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: Association for Computing Machinery, 2007, pp. 29–42. ISBN: 9781595939081. DOI: 10.1145/1298306.1298311. URL: https://doi.org/10.1145/1298306.1298311 (cit. on p. 42).

[93] A. Mislove et al. “Growth of the Flickr Social Network”. In: *Proceedings of the First Workshop on Online Social Networks*. New York, NY, USA: Association for Computing Machinery, 2008, pp. 25–30. ISBN: 9781605581828. DOI: 10.1145/1397735.1397742. URL: https://doi.org/10.1145/1397735.1397742 (cit. on p. 42).

[94] C. Wilson et al. “User Interactions in Social Networks and Their Implications”. In: *Proceedings of the 4th ACM European Conference on Computer Systems*. New York, NY, USA: Association for Computing Machinery, 2009, pp. 205–218. ISBN: 9781605584829. DOI: 10.1145/1519065.1519089. URL: https://doi.org/10.1145/1519065.1519089 (cit. on p. 42).

[95] D. Achlioptas et al. “On the Bias of Traceroute Sampling: Or, Power-law Degree Distributions in Regular Graphs”. In: *Journal of the ACM* 56.21 (4 2009), pp. 1–28. DOI: 10.1145/1538902.1538905 (cit. on p. 42).

[96] N. M. Amato and Y. Wu. “A randomized roadmap method for path and manipulation planning”. In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 1. 1996, pp. 113–120. DOI: 10.1109/ROBOT.1996.503582 (cit. on p. 43).

[97] M. H. Overmars. *A Random Approach to Motion Planning*. Tech. rep. 1992 (cit. on p. 43).

[98] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. ISSN: 0029-599X. DOI: 10.1007/BF01386390 (cit. on pp. 43, 54, 55).

[99] S. M. LaValle and Jr. J. J. Kuffner. "Randomized Kinodynamic Planning". In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400. DOI: 10.1177/02783640122067453. URL: https://doi.org/10.1177/02783640122067453 (cit. on p. 44).

[100] D. González et al. "A Review of Motion Planning Techniques for Automated Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (2016), pp. 1135–1145. DOI: 10.1109/TITS.2015.2498841 (cit. on p. 44).

[101] S. M. Lavalle. "Rapidly-Exploring Random Trees: A New Tool for Path Planning". In: (1998) (cit. on pp. 44, 101).

[102] S. Karaman and E. Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. ISSN: 0278-3649. DOI: 10.1177/0278364911406761. URL: https://doi.org/10.1177/0278364911406761 (cit. on pp. 44, 45).

[103] A. Ivanov and M. Campbell. "An efficient robotic exploration planner with probabilistic guarantees". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4215–4221. DOI: 10.1109/ICRA.2016.7487616 (cit. on p. 44).

[104] H. Umari and S. Mukhopadhyay. "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 1396–1402. DOI: 10.1109/IROS.2017.8202319 (cit. on p. 44).

[105] S. M. LaValle. *Planning Algorithms*. 1st. Cambridge: Cambridge University Press, 2006, p. 842. URL: http://msl.cs.uiuc.edu/planning/ (cit. on pp. 44, 45).

[106] P. Cheng, Z. Shen, and S. M. Lavalle. "RRT-Based Trajectory Design for Autonomous Automobiles and Spacecraft". In: *Archives of Control Sciences* 11.3-4 (2001), pp. 167–194. DOI: citeulike-article-id:4232681 (cit. on pp. 45, 80).

[107] K. Anand and L. Devroye. "Probabilistic Analysis of RRT Trees". In: *arXiv preprint arXiv:2005.01242* (2020) (cit. on pp. 45, 63).

[108] L. Ma et al. “A fast RRT algorithm for motion planning of autonomous road vehicles”. In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 2014, pp. 1033–1038. ISBN: 2153-0017. DOI: 10.1109/ITSC.2014.6957824 (cit. on pp. 45, 63).

[109] J. G. Kang et al. “Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning”. In: *Sensors* 21.2 (2021). ISSN: 1424-8220. DOI: 10.3390/s21020333 (cit. on pp. 45, 63).

[110] F. Islam et al. “RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution”. In: *2012 IEEE International Conference on Mechatronics and Automation*. 2012, pp. 1651–1656. DOI: 10.1109/ICMA.2012.6284384 (cit. on p. 46).

[111] L. E Dubins. “On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents”. In: *American Journal of mathematics* 79.3 (1957), pp. 497–516 (cit. on p. 46).

[112] Mišel Brezak and Ivan Petrović. “Real-time approximation of clothoids with bounded error for path planning applications”. In: *IEEE Transactions on Robotics* 30.2 (2013), pp. 507–515 (cit. on p. 46).

[113] A. Ravankar et al. “Path Smoothing Techniques in Robot Navigation: State-of-the-Art, Current and Future Challenges”. In: *Sensors (Basel, Switzerland)* 18.9 (2018), pp. 31–70. DOI: 10.3390/s18093170 (cit. on p. 46).

[114] K. Komoriya and K. Tanie. “Trajectory design and control of a wheel-type mobile robot using B-spline curve”. In: *Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems'. (IROS'89)'The Autonomous Mobile Robots and Its Applications*. IEEE. 1989, pp. 398–405 (cit. on p. 46).

[115] S. X. Yang and M. Meng. “An efficient neural network approach to dynamic robot motion planning”. In: *Neural networks* 13.2 (2000), pp. 143–148. ISSN: 0893-6080 (cit. on p. 47).

[116] E. A Antonelo and B. Schrauwen. “Supervised learning of internal models for autonomous goal-oriented robot navigation using reservoir computing”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2959–2964. ISBN: 1424450381 (cit. on p. 47).

[117] S. Kim and B. An. “Learning Heuristic A: Efficient Graph Search using Neural Network”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9542–9547. DOI: 10.1109/ICRA40945.2020.9197015 (cit. on p. 47).

[118] L. Breiman et al. *Classification and regression trees*. Routledge, 2017 (cit. on p. 48).

[119] M. Pfeiffer et al. "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1527–1533. DOI: 10.1109/ICRA.2017.7989182 (cit. on p. 48).

[120] S. P. Singh et al. "Robust reinforcement learning in motion planning". In: *Advances in neural information processing systems*. 1994, pp. 655–662 (cit. on p. 48).

[121] Y. Li, C. Li, and Z. Zhang. "Q-Learning Based Method of Adaptive Path Planning for Mobile Robot". In: *2006 IEEE International Conference on Information Acquisition*. 2006, pp. 983–987. DOI: 10.1109/ICIA.2006.305871 (cit. on p. 48).

[122] G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, 1963 (cit. on p. 54).

[123] S. E. Dreyfus. "An Appraisal of Some Shortest-Path Algorithms". In: *Operations Research* 17.3 (1969), pp. 395–412. ISSN: 0030364X, 15265463. URL: http://www.jstor.org/stable/168375 (cit. on p. 54).

[124] R. Floyd. "Algorithm 97: Shortest path". In: *Communications of the ACM* 2 (1962), p. 345 (cit. on p. 54).

[125] T. H. Cormen et al. *Introduction to Algorithms, Third Edition*. The MIT Press, 2009. ISBN: 0262033844 (cit. on pp. 55, 61).

[126] J. R. Evans and E. Minieka. *Optimization Algorithms for Networks and Graphs*. Marcel Dekker, New York, 1992 (cit. on p. 55).

[127] L. Zhang et al. "Rapidly-exploring Random Trees multi-robot map exploration under optimization framework". In: *Robotics and Autonomous Systems* 131 (2020), p. 103565. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2020.103565. URL: http://www.sciencedirect.com/science/article/pii/S092188902030405X (cit. on p. 63).

[128] R. L. Sproull. "Refinements to nearest-neighbor searching in k-dimensional trees". In: *Algorithmica* 6 (1991), pp. 579–589 (cit. on p. 63).

[129] E. Plaku, L. E. Kavraki, and M. Y. Vardi. "Motion Planning With Dynamics by a Synergistic Combination of Layers of Planning". In: *IEEE Transactions*

on Robotics 26.3 (2010), pp. 469–482. ISSN: 1941-0468. DOI: 10.1109/TRO.2010.2047820 (cit. on p. 63).

[130] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 2997–3004. ISBN: 2153-0866. DOI: 10.1109/IROS.2014.6942976 (cit. on pp. 63, 102).

[131] C. D. Toth, J. O'Rourke, and J. E. Goodman. *Handbook of Discrete and Computational Geometry*. CRC press, 2017 (cit. on p. 75).

[132] D. C. Conner, A. A. Rizzi, and H. Choset. "Composition of local potential functions for global robot control and navigation". In: *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. Vol. 4. 2003, pp. 3546–3551. DOI: 10.1109/IROS.2003.1249705 (cit. on p. 80).

[133] A. Dürer. *Underweysung der Messung, mit dem Zirckel und Richtscheyt, in Linien, Ebenen unnd gantzen corporen*. Nüremberg: Hieronymus Andreae, 1525 (cit. on p. 84).

[134] C. Leopold. "Geometry and Aesthetics of Pentagonal Structures in the Art of Gerard Caris". In: 2016 (cit. on p. 84).

[135] R. Penrose. "Pentaplexity A Class of Non-Periodic Tilings of the Plane". In: *The Mathematical Intelligencer* 2 (1979), pp. 32–37. DOI: 10.1007/BF03024384 (cit. on p. 84).

[136] A. Dürer. *Underweysung der Messung, mit dem Zirckel und Richtscheyt, in Linien, Ebenen unnd gantzen corporen*. Nüremberg: Hieronymus Andreae, 1525 (cit. on p. 83).

[137] R. Penrose. "Pentaplexity A Class of Non-Periodic Tilings of the Plane". In: *The Mathematical Intelligencer* 2 (1 1979), pp. 32–37. DOI: 10.1007/BF03024384 (cit. on p. 83).

[138] B. Grunbaum and C. G. Shephard. *Tilings and Patterns*. Freeman, NewYork, 1986 (cit. on p. 83).

[139] S. S. Sastry and Z. Li. "Robot motion planning with nonholonomic constraints". In: *Proceedings of the 28th IEEE Conference on Decision and Control*. Vol. 1. 1989, pp. 211–216. DOI: 10.1109/CDC.1989.70105 (cit. on p. 100).

[140] Z. Li, J. F. Canny, and S. S. Sastry. "On motion planning for dexterous manipulation. I. The problem formulation". In: *International Conference on Robotics and Automation*. 1989, 775–780 vol.2. DOI: 10.1109/ROBOT.1989.100078 (cit. on pp. 100, 103).

[141] A. Marigo and A. Bicchi. "A local-local planning algorithm for rolling objects". In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. 2002, pp. 1759–1764. DOI: 10.1109/ROBOT.2002.1014796 (cit. on p. 100).

[142] D. Peng and K. J. Hanley. "Contact detection between convex polyhedra and superquadrics in discrete element codes". In: *Powder Technology* 356 (2019), pp. 11–20. ISSN: 0032-5910. DOI: https://doi.org/10.1016/j.powtec.2019.07.082. URL: https://www.sciencedirect.com/science/article/pii/S0032591019305741 (cit. on p. 100).

[143] S. C. Jacobsen et al. "TheUtah/MITHand:Work in Progress". In: *International Journal ofRobotic Research* 4.3 (1984), pp. 21–5– (cit. on p. 101).

[144] N. T. Lam, I. Howard, and L. Cui. "Path planning for the Platonic solids on prescribed grids by edge-rolling". In: *PLOS ONE* 16.6 (2021). DOI: 10.1371/journal.pone.0252613 (cit. on p. 101).

[145] A. Bloch, J. Marsden, and D. Zenkov. "Nonholonomic Dynamics". In: *Notices of the AMS* 52 (2005) (cit. on p. 103).

[146] A. Bicchi and R. Sorrentino. "Dexterous manipulation through rolling". In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Vol. 1, pp. 452–457. ISBN: 1050-4729. DOI: 10.1109/ROBOT.1995.525325 (cit. on p. 103).

[147] S. A. Chaplygin. "On some feasible generalization of the theorem of area, with an application to the problem of rolling spheres". In: *Mat. Sbornik* 20 (1897), pp. 1–32 (cit. on p. 103).

[148] S. A. Chaplygin. "On a rolling sphere on a horizontal plane". In: *Mat. Sbornik* 24 (1903), pp. 139–168 (cit. on p. 103).

[149] A. B. A. Cole, J. E. Hauser, and S. S. Sastry. "Kinematics and control of multifingered hands with rolling contact". In: *IEEE Transactions on Automatic Control* 34.4 (1989), pp. 398–404. ISSN: 0018-9286. DOI: 10.1109/9.28014 (cit. on p. 103).

[150] R. Rhodes and Dark S. *The Making of the Hydrogen Bomb*. New York: Simon & Schuster: Touchstone Books, 1999, p. 731 (cit. on p. 106).

[151] H. W. Kroto et al. *C60: Buckminsterfullerene*. Vol. 318. Nature, 1985, pp. 162–163. DOI: https://doi.org/10.1038/318162a0 (cit. on p. 106).

[152] J. Itoh, J. O'Rourke, and C. Vîlcu. "Star unfolding convex polyhedra via quasigeodesic loops". In: *Discrete & Computational Geometry* 44.1 (2010), pp. 35–54 (cit. on p. 110).

[153] E. Miller and I. Pak. "Metric Combinatorics of Convex Polyhedra Cut Loci and Nonoverlapping Unfoldings". In: *Discrete & Computational Geometry* 39 (2008), pp. 339–388. DOI: https://doi.org/10.1007/s00454-008-9052-3 (cit. on p. 110).

[154] S. Karaman and E. Frazzoli. "Incremental Sampling-based Algorithms for Optimal Motion Planning". In: *CoRR* abs/1005.0416 (2010). arXiv: 1005.0416. URL: http://arxiv.org/abs/1005.0416 (cit. on p. 124).