School of Design and Built Environment

Form to Behaviour: Rethinking Group Form

Andrei Smolik

This thesis is presented for the degree of
Master of Philosophy
Of
Curtin University

April 2021

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of the work which has been carried out since the official commencement date. Any editorial work, carried out by a third party is acknowledged. Ethics procedures and guidelines have been followed.

Signed: _____

Andrei Smolik

April 23rd 2021

Thesis Supervisor: Professor Sambit Datta

Title: Professor of Architecture, School of Design and Built Environment, Faculty of Humanities, Curtin University

Thesis Co-supervisor: Dr. Boon Lay Ong

Title: Senior Lecturer, School of Design and Built Environment, Faculty of Humanities, Curtin University

Acknowledgements

I would firstly like to thank Professor Sambit Datta for facilitating this thesis by providing intellectual input and funding for projects, conferences and workshops. Without his support this thesis would not be possible. Another person I would like to thank is Robert Cameron, a friend and collaborator whose passion and commitment encouraged me to keep developing my skills as a designer and coder. His input on *Cellular Form* and *Mediated Form* projects could not be overstated, both in terms of work spent on constructing these prototypes as well as helping me develop skills through collaborative practice. Big thanks has to go to Professor Teng Weng Chang of the Yun Lin University and everyone at the Idea Factory who facilitated me during the *Resonance* workshop and development of work in this thesis that eventuated in the construction of the *Kinetic Form* project in Shenzhen. Their development of digital fabrication facilities and focus on interaction design helped shape some of the ideas presented here. Also big thanks to friends and family who supported me during the writing of this thesis, your love and support was vital.

Definitions and Abbreviations

Workflow	Is a coordinated and modular pattern of design organisation enabled by systematic arrangement of materials and resources into processes that creates more complex systems and organisations.
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CNC	Computer Numeric Control
Mesh	A computer graphics model for representing shapes of polyhedral objects in 3D
NURBS	Non-Uniform Rational B-Splines- a mathematical model commonly used in computer graphics to represent curves and surfaces
LOD	Level of Detail

Abstract

This thesis investigates form and system--generating processes in architecture in the context of design computation, system design and digital fabrication. It is based on the idea of metabolism- formulation of architecture as a living, evolving and expanding network of components operating as a system, rather than a static object first introduced by Japanese architects in the early 60's as a design strategy for large scale urban interventions. This research expands this notion to include architecture at smaller scales in light of contemporary advances in computer aided design, digital fabrication and interactive design. Thus, the research attempts to explore ideas of architecture as 'inspired' by nature and incorporate generative principles directly into its design, fabrication and operating behaviour.

The aim of this research is to develop understanding of methods and techniques for the development and fabrication of the built environment based on metabolic principles- that is principles, which are representative of natural systems. This methodology could theoretically be applied to any collection of parts at larger scales and will enable architects and designers to develop new models of architecture that could bring about improvements in the built environment in the future.

The objectives of the research are three-fold. Firstly, it aims to investigate design of aggregations in group-form both top-down and bottom-up. Second, it investigates approaches to design of architectural responsiveness in the context of formal metabolic approaches. Third, it explores combinatorial approaches to applying form, interactive responsiveness and data fields to group-form approach.

Primary investigation and evaluation is accomplished through design of three experimental architectural prototypes: *cellular form, mediated form* and *kinetic form*. Each focuses on separate research area: bottom-up generation of form through developing of components with aggregative logic, using physics simulation in the design process and developing complex fabrication workflows and finally, creating a responsive kinetic system made out of component parts.

All three investigations were completed as part of public exhibitions and fairs allowing the public to experience the space and interactivity aspects of each installation. This was done to demonstrate potential applications of the ability to rapidly construct these objects out of discrete parts and test their ability to sense real-time data and give feedback to the users, or change their formal configuration in response.

Contents

1. Introduction

More than fifty years since the *Investigations and Linkage in Collective Form* proposed a reconceptualization of architectural approaches to large urban forms and a methodological shift from "master plan" to "master program" suggesting a temporal/ operational consideration of the design object. Today we have a different set of tools available to us with which we may continue exploration of this concept from a different perspective. Practice methodologies of both urban and architectural design are today in the process of restructuring to accommodate new approaches that take advantage of automation of construction, fabrication and assembly and new digital simulation and modelling tools that allow for more precise and informed descriptions. However, there still exists a gap between modular mass-production construction and fabrication industry and the interests of the architectural practice whose interests lie in the domain of pure design.

This research is set out to interrogate emerging practices within architecture, informed by the theory of group form as well as contemporary techniques of conventional and non-conventional material and digital experimentation with simulation, digital fabrication, assembly and prototyping. The focus is on exploring approaches to design enabled by modern tools of conceptualisation, communication and realisation and informed by Maki's theory of *group form* which elaborates on the ideas of metabolism as a growth of built form (Maki, 1962). Emerging out speculative ideas to design assemblies attempting to deal with complexities of building clusters and even cities, this thesis will instead focus on small-scale design experimentation and will demonstrate how this design theory finds a broader application to a wider range of scales through analysis of organisation of workflows across design and fabrication.

This research is driven by design project work incorporating in its investigation a broad domain across digital fabrication, simulation and interaction design. Three prototypes are created alongside various tools and techniques. Group form is reconceptualised here as a set of conceptual and practical applications that begin to enable a designer to work with form and program simultaneously through the effects of organisation. With this new understanding of contemporary methods of design computation, this thesis attempts to re-examine ideas initiated with Maki by changing the focus from large scale urban problems to a more theoretical and conceptual problems of design process through construction of small-scale prototype design assemblies tracking and developing on ideas of *group form*. This is done for a couple of reasons. First, it allows rapid prototyping of prototypes at compressed time scales allowing for a more immediate assessment of built assemblies that are relatively cheap and simple to construct. Second, the application and theory is more generalised-

architectural assembly of collective form no longer operates at a specific scale but is seen as having agency across multiple functions and scales. This allows it to transcend conventional typologies of parts associated with urbanism and focus more on compositional and methodological strategies. Thus, the research is informed by two overarching themes or objectives that are in close dialogue with each other: one is the development and exploration of digital tools and techniques for architectural design; the other is its implications on design strategies of group form as a system of simultaneous form and program organisation. The first objective is comparable to engineering methodology of formulating hypothesis and testing precise routines, the latter is a cultural endeavour of defining the design agenda and speculating on the consequences of technological development.

1.1. Motivation

The focus is on exploring approaches to design enabled by modern tools of conceptualisation, communication and realisation and informed by Maki's theory of *group form* which elaborates on the ideas of metabolism as a growth of built form (Maki, 1962).

The design exploration of *group form* described in this thesis attempts to demonstrate a series of practical and conceptual tools, which enable designers to work on form and program simultaneously through the agency of organisation. This organisation is itself modular- composed of clusters of components that mediate information exchange between descriptions of design objects, their behaviour and methods of their construction and organisation. This shift from linkages of form to linkages of information exchange is counterintuitive but conceptually very powerful. It echoes practices that deal with complex interacting systems in both scientific disciplines such as climate and economy and engineering disciplines such as software design.

1.2. Research Question

Technological advances in the field of computer-aided architecture presents new opportunities for designers and architects. Through construction of parametric models and simulations, designers are now able to take advantage of 3D modelling and scripting tools in order to construct iterative, testable solutions. These solutions can be driven by both designer input parameters, data collected from networks, existing information, models of real-world sensor readings or iterated through a solution space given a certain set of optimisation criteria. These methods and tools have been extensively used in contemporary architectural research and practice with theoretical aspirations

often looking towards precedents in biology to create theoretical frameworks for architecture. This has often resulted in architecture that is focused extensively on engineered visual effects created by manipulation of geometry and materials at limited scales and with applicability that is restricted to specific material implementation and typology.

The thesis attempts to make a connection between the tool-centred approaches of contemporary architectural inquiry with ideas developed in from the 1960's beginning with Maki's group form and connected to various later developments in algorithmic design and digital fabrication. It is argued here that this approach could be generally considered a variant of the bottom-up design principle as it focuses primarily on design and distributions of interconnecting elements in space but unlike some modern conceptions of a bottom-up design methodology, in this variant, the author maintains creative control as opposed to democratising creative decision-making. Loosely defined as 'group form' or 'collective form' by Maki, it gained a more specific sets of characteristics drawing from existing settlements and vernacular villages as a starting point, thus acquiring more sociological and political contexts. Even though, the original aspirations of metabolism were development of a theory of multiple scales and resolutions (Noboru, 1960), the primary focus has remained on urban-scale with many specific built examples by the group remaining largely within conventional framework of modernism.

The society for which Metabolist architecture was envisioned has changed radically, acquiring new links to the technologies that help produce architecture, connect people digitally and, in some cases, directly interact with them. The aforementioned methods of producing architecture have also changed, taking advantage of accessible visualisation, 3D modelling and scripting tools allowing for new possibilities previously unavailable to Maki. Given that the ideas of group form are deeply rooted in architecture's internal logic that was envisioned to expand and respond to dynamic environments and events, the key aim of this thesis is to advance abstract concepts articulated as group form into the realm of contemporary digital computer-aided design approaches. Based on the aims outlined above, this thesis asks the following question:

> How can computer-aided design, construction, its visualisation, its physical construction and interaction with people be used to create architecture based on group form principles?

This key question leads to a series of project-based design investigations in conjunction with literature and precedent review. The underlying hypothesis is that through series of project works and practices, it would be possible to test the limits and possibilities of applying emerging scripting, digital fabrication and interaction to principles articulated fifty years ago and tested primarily at a different scale. These investigations serve as test beds to experiment with variety of techniques and

approaches in order to tackle specific sub-goals. They were developed to understand implementation of specific approaches to particular research areas and conceived as an iterative process of 'reflection-in-action' where assessment of creative output is incorporated into a feedback loop which contributes to a bridge between theory and practice (Schon, 1983).

It is important to note that this research does not attempt to re-create architecture of Metabolists through different means or create perfect technological implementation of these concepts. Rather, it seeks to establish an updated conceptual framework and allow results to emerge at the conclusion of these investigations. The outcomes are qualitative. By developing a consistent experimental framework in which to design, prototype and evaluate design applications through novel fabrication methodologies and tools, a new design approach for group form concept can be realised. This hypothesis is assessed by the results generated from the design investigations, as ideas develop through operational digital and physical conceptual prototypes. By suggesting experimental design methods, refining and testing both digital and physical workflows, this research provides architects and designers with a combination of practical and conceptual design tools to apply group form systems in the early-stage design of architecture at varying scales.

This research, therefore, expands on the concepts of group form by deploying it in a different context and applying new digital tools of parametric design, fabrication and physical/digital interaction through presented investigation process and prototypical design outcomes.

1.3. Structure of Exegesis

This exegesis is divided into seven main parts and structured around three main design investigations in chapters four, five and six. Chapter 1. Introduction serves as a general outline of the research question and hypothesis, its aims and methodology. It outlines the parameters of investigations that were conducted in this thesis and the manner of analysing its results. Chapter 2. : Background presents all relevant background material and constitutes exegesis' literature review. It is broken down into the examination of the concept of *Metabolism* in architecture, *System Design* serves as a review of computational architectural design and its brief history, *Computational Approaches to Form Aggregation* examines parallels between the process of design, its naturalistic precedents and the tools that enable their implementation, and *Responsive Architecture* reviews how system design manifests in architecture and helps designers to approach re-configurable assemblies.

The body of exegesis consists of three main design investigations: Design Investigation 1: *Cellular Form*, Design Investigation 2: and, finally Design Investigation 3: *Kinetic Form*. Each investigation is

further broken down into four subchapters which serve as descriptive sections, tracking design and construction of these prototypes as follows:

- Section one describes conceptual design process always incorporating system design approach- a script or a procedure is proposed and elaborated upon which stems from a broader thesis of aggregate investigation in design process.
- Section two of each Design Investigation investigates material and tectonic fabrication process of each prototype and how scaled construction imposes constraints and presents new opportunities for physical assembly process.
- Section three of each Design Investigation describes interaction of each prototype and how its systems interface with users and integrate into the physical object.
- Section four and five are summary and discussion- they serve to conclude each Design Investigation chapter by analysing results of the design prototyping process and provide an overview of the developing argument.

Chapter 7 provides overall discussion on how concepts presented in Research Aim have been achieved, which methodologies proved most successful in achieving these outcomes and potential for future investigations. Chapter 8. Conclusion provides a discussion of the relationship between aggregative form and methods for its construction and implementation of its systems in future research and design applications.

2. Background

Group form is a term coined by Fumihiko Maki to describe a novel compositional and analytical approach to architecture developed out of the Metabolist movement of which Maki was a key member. Originating from reflections on the work of the Metabolists and embodying a type of contextual theoretical development of Modernism, its legacy can be traced in both architectural theory and practice from built works of Rem Koolhaas to Stan Allen's concept of 'field conditions'. On one hand, group form requires a theoretical update considering a radically new technological context for both production and conception of architecture that since the time of publication of *Investigations* developed from a rhetorical evocation of systems to a somewhat more literal application of algorithms and parametric design approaches in ever-increasing aspects of design and construction. On the other hand, modern technologies suggest a more direct possibility of metabolic processes possible through responsiveness to context and patterns of habitation, on the other it offers even more exotic possibilities of direct self-assembly and growth as described by one of the forerunners of contemporary computer-aided branch of architectural research, John Frazer. Today, this general architectural research field expanded to include advanced architectural geometry, simulation, robotic fabrication, IOT (Internet of Things), interactive architecture and many other technologically enhanced enquiries of similar kinds. These often have a narrow focus on primary application to very specific technologies of architectural production. Not unlike Maki, CAD forerunners' vision for architecture is much broader, generally all-encompassing theories that have the potential to contribute to all aspects of architectural production including the way we conceive and think about architecture from first principles as well as a complex intersection of structural, utilitarian, ecological, phenomenological, social, political and cultural concerns.

2.1. Architectural Composition

One of the most prominent themes in Architectural theory, in which group form is firmly positioned, is the composition or how architecture can be put together. As mentioned earlier, Maki's contribution was to introduce the problem of indefinite scaling and how architecture can accommodate ever-expanding scales through focus on process and radical modularity. More recent prominent concept that displays certain conceptual similarity is field conditions, developed by Stan Allen in the early 1990's. He describes it as *"any formal or spatial matrix capable of unifying diverse elements while respecting the identity of each. Field configurations are loosely bounded aggregates characterized by*

porosity and local interconnectivity…What is intended here is a close attention to the production of difference at the local scale, even while maintaining a relative indifference to the form of the whole." (Allen, 1997, p. 24). Developed largely from ideas in other fields, including music and art, field conditions borrow elements from *processes*, movements and generative instructions to suggest an approach to making architecture that stands in opposition to previous compositional approaches such as modernistic grid-based approaches like those of Mies Van der Rohe or the more chaotic and expressionist strategies of deconstructivism (Allen, 2011).

Joseph Becker, a MOMA curator of exhibition titled Field Conditions traces the origins of this concept to developments in art in the twentieth century. He brings up Jackson Pollock and the term 'allover' to describe decentralised and 'polyphonic' composition that relies on closely similar elements which *"repeat themselves without marked variation from one edge of the picture to the other"* (Becker, 2012). This allows painting to achieve a separation between the subject, the object and thus, to dissolve the space between. Another important development in relation to field conditions is the minimalist and process art of the 1960's. Of particular interest is the work of Sol LeWitt who, similar to Pollock, developed work that dissolved decipherable form. Unlike Pollock, LeWitt's approach can be called emergent or procedural- the 'art object' is executed by assistants through instructions (Figure 1). This leads to an aesthetic of modularity and uncertainty about the outcome which develops into the technique of developing processes rather than objects or drawings.

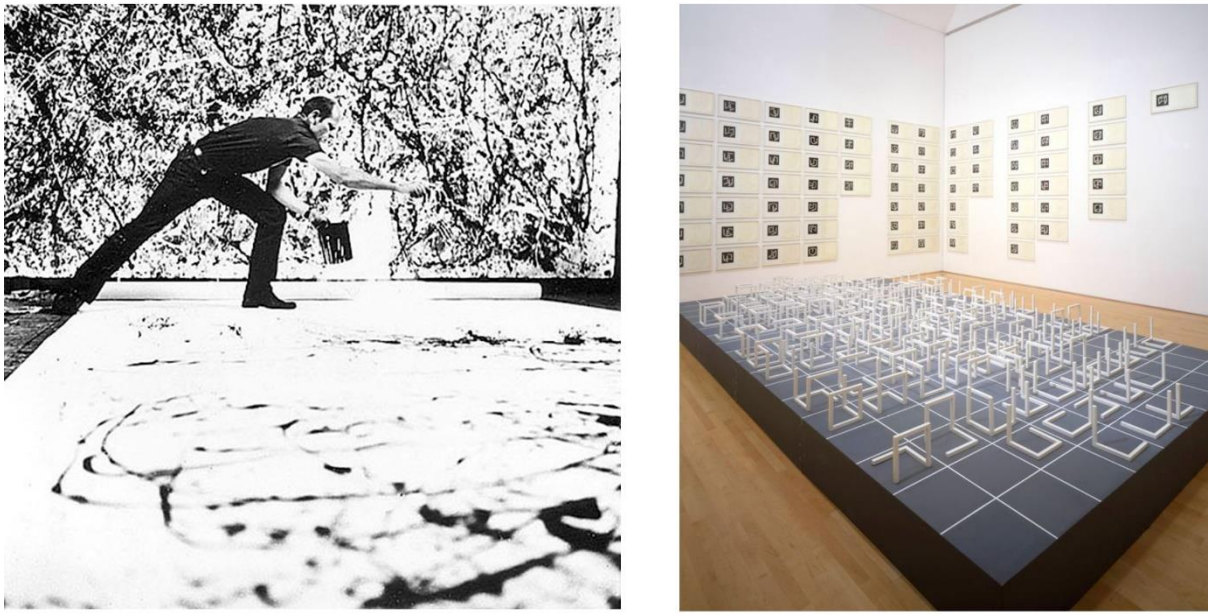


Figure 1: Left: Jackson Pollock, 1950, right: Sol LeWitt, Incomplete Open Cubes, 1974

Contemporaries of the Metabolists, the work of avant-garde pioneers like Constant, Archigram, Superstudio, and Archizoom illustrated abstract architectural worlds expanding in scale as long-term structures that would support short-term components. Particularly in the case of Archigram, various projects that included drawings such as that of a Computer City (made long before the widespread of computer technologies in 1964- see Figure 2) suggested a "system approach", which would enable the type of sprawling, component-based aggregated design of a "Plug-In City" to operate through the use of monitoring systems, radio-controlled taxis, ambulance services, and airports (Sadler, 2005, p. 20).

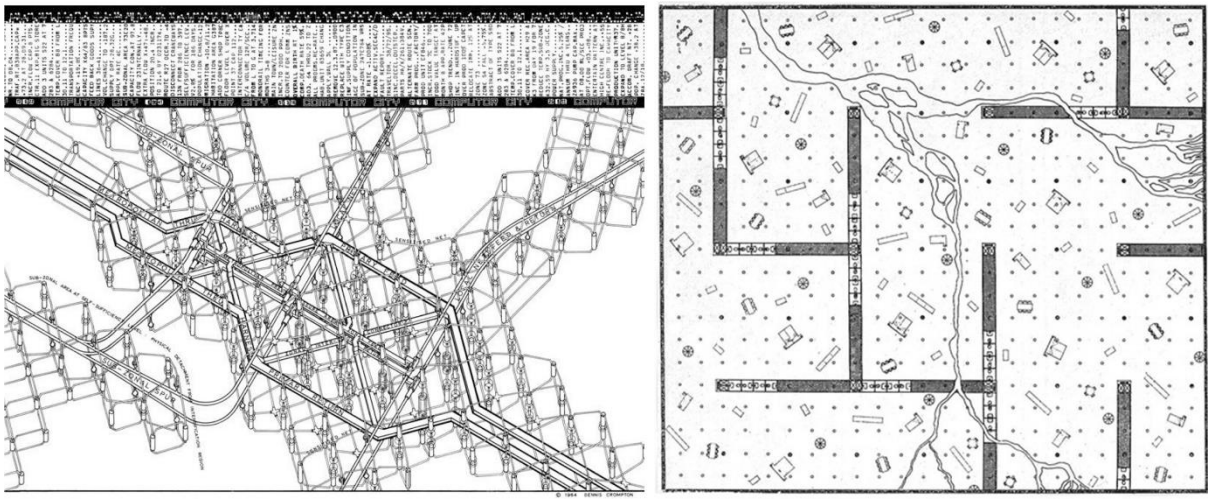


Figure 2: Left: Dennis Crompton, Computer City, 1964, Right: Archizoom's No Stop City, 1971

2.2. Group Form

Group-form is a term idea developed by Fumihiko Maki while he was a member of the faculty at the School of Architecture at Washington University from 1956 to 1963. The work, made of two main parts: *Investigations in Collective Form* and *Linkages in Collective Form* attempts to describe a methodological framework for creation of what Maki calls *group form*- a particular configuration or assemblage that *'evolves from a system of generative elements in space'* (Maki, 1964, p. 14). The agenda is outlined in the opening paragraphs as an open-ended question, seeking a set of strategic tools for the making of the physical environment. The main criticism is centred on composition of larger, open-ended urban forms which, in Maki's view, lacks elasticity and flexibility as well as visual and physical presence coherent with the function and technology which gives rise to it. Thus, the

Collective Form is an attempt at formulating a design approach for creation of assemblages beyond single objects or formal compositions.

Maki's concept of group form first emerged in the essay titled "Towards Group Form", co-authored with Masato Ohtaka and featured in *Metabolism/1960: The Proposals for New Urbanism* (Noboru, 1960). Four years later, Maki publishes *Investigations in Collective Form* which elaborates further on this idea (Maki, 1964). Conceptually, he suggests three distinct categories for looking at architectural composition rooted in different approaches to part-to-whole relationships. *Compositional Form* is described as the traditional approach for building design that completes a formal statement-individual buildings are made from components arranged to complete a formal statement as well as to satisfy functional, visual and spatial relationships. *Investigations in Collective Form* is the first written work to define the idea of *megastructure* as a frame which houses modular components comprising of a diverse set of programs concentrated in a single composition. Made possible by modern technology, Maki suggests that it allows the creation of an artificial infrastructural landscape which would allow for different metabolic cycles to emerge within its boundaries. This category, in fact, is characteristic of the metabolic movement with Kurokawa's *Agricultural City* and *Helix City*, Isozaki's *City in the Air* and *Joint Core System*, Tange's *Tokyo Bay Plan* all deploying the megastructure approach (Zhongjie, 2010, p. 112).

Group Form (or Collective Form) is presented as an alternative to megastructure which is critiqued for its "certain static nature" inherent to it. Instead of providing a physical structure, group form presents a master program or grammar which allows for coping with expanded scales, complexity and transformations. Maki's distinction between 'form' and 'system' is based on Louis Khan's ideas of 'form' and 'design' presented at the World Design Conference in 1960 (Maki, 1962, p. 20).

This category begins to suggest that the systemic property of mass program is inherent in parts composing the group: a composition which "evolves from a system of generative elements in space" Maki's *Linkage in Collective* Form attempts to formulate a design strategy around group form:

Collective form also requires a new dimension in conceiving construction methods and structural and mechanical systems. The aesthetics of the collective form necessitate new definitions of scale and proportion of buildings.

Above all, this entire essay questions the meaning of the very act of "design" in our society.[=] (Maki, 1964, p. 28)

As a designer, Maki is concerned with operation application of collective form. In the section on *operational categories*, he uses examples to identify abstract principles governing group form. This is

of a particular interest to this thesis, as we are concerned primarily with the compositional essence of group form and its constraints rather than its specific urban application. From these abstract principles, application is made possible at a more generalizable scale as Noboru Kawazoe once made a similar observation in regards to Metabolism as being *"a continuous development from atom to nebula"* (Noboru, 1960, p. 5).

"Ultimately we seek new ground that inexorably links process and product: the former constantly structures and restructures the letter in a feedback loop." (Mayne, 2011, p. 12)

Similarly, Stan Allen[1] in his essay *Field Conditions* articulates a broad bottom-up design theoretical model which draws on architectural precedents such as the Great Mosque of Cordoba and Le Corbusier's 1964 Venice Hospital proposal; shifts from minimalism to post-minimalism in contemporary art with its emphasis on processes; new compositional techniques in music based on crowd behaviour like the Xenakis's *Metastasis* and; finally, flocking behaviour developed in computer science by Craig Reynolds in the 1980's (Allen, 1999). Similarly to Maki's group form, field conditions is built environment conceived of as a series of parts beginning with a single element or elements that cluster or aggregate to become many. Allen is interested in the logic of aggregation and the effects of that logic and the potential dynamic nature of the assembly: *"More than a formal configuration, the field condition implies an architecture that admits change, accident and improvisation*" (Ibid., 101). This is evident in his diagrams that speculate on various effects of, for example, moire patterns that echo some operational categories from Maki. However, Allen did not develop his approach from Maki, but instead conceived of it as a critique of deconstructionist movement in architecture which attempted to move towards 'additive' approaches as opposed to the prevailing 'subtractive' ones.[2] In the preface to *Combinatory Urbanism*, Allen acknowledges an 'uncanny familiarity' of Maki's ideas on group form and its shift from form to behaviour under the domain of complexity: *"[Maki] recognises that the city is a dynamic mix of program, politics and activity but understands that architecture's primary agency is to define form, now in new configurations that promote rather than restrict change over time."*[3]

Allen's contribution to the idea of group form/ field conditions is its extension into a much more complex organisational principle that incorporates art, architecture, politics, psychology and

[1] Stan Allen is an American architect, theorist and former dean of Princeton University's School of Architecture

[2] (Allen, 2011) Allen discusses context behind the essay in his conversation with Scott Cohen at GSD: *"We were all reacting against deconstructivism. So the model of deconstructivism which was that you started with the whole figure and you broke it apart. Think of Schumi and disjunction, discontinuity and Peter Eisenman starting with the whole figures and fracturing them in different ways"*.

[3] (Mayne, 2011, p. 57) is a collection of urban projects by Morphosis Architects from 1999-2006 which contains project descriptions, urban analysis studies and architecture theory essays.

technology. This thesis constrains the scale of the exploration of this approach and instead attempts to expand application of systems (or digital design approaches) to the study of group form and its composition.

2.3. Computational Design & Prototyping

One of the key developments that animated a resurgence in the application of biological systems and processes in architecture and design in the last twenty to thirty years is a wider availability of scripting tools. This brings with it certain questions in regards to how these new tools are applicable to design processes:

In looking outside architecture for some clues on how we might do a better job with performance, it is hardly surprising that we therefore seek guidance from nature, there being no other 'system' available for us to study for convincing prototypes. The intellectual migration for architects thinking from object-related thinking (building form) to one of systems (building performance) is well on its way, and scripting naturally offers a means to queue jump millions of years of evolution, by computationally filling in for time's selection of the fittest solutions (Burry, 2011, pp. 233-234).

Mark Burry[4] is referring to a narrow design process (evolutionary algorithm) with a focus on specific outcome (performance). Defining performance for architecture is a daunting task since it not only encompasses a very large set of interrelated performative parameters (form, structure, usage etc.) but also aspects that cannot easily be addressed by any formal system or description (aesthetics, design creativity, style).

These new tools and approaches allow for novel ways of investigating group form. Specifically, the impact of digital fabrication on rapid deployment of customisable, adaptable components at localised scales and the use of interactive information technologies to allow for communication and adaptability of form given particular stimulation in the process of programming architecture as a responsive system.

[4] Mark Cameron Burry is the Professor of Urban Futures at the Melbourne School of Design at The University of Melbourne.

3. Research Methodology

In the previous chapter I have framed a context for exploring combinatory form through digital fabrication. This encompasses scripting and simulation, material and form as well as fabrication and performance. I have discussed conceptual origins of combinatory form as well as practices that relate to workflows in architecture that begin to explore ideas of modularity, system design and provisional morphology capable of *'spontaneous experience and ad hoc formulations'* (Mayne, 2011, p. 36).

The work in this research is undertaken as a strategy of reflection-in-action[5]. This involves iterative design, fabrication and critical reflections on outcomes. The prototypes that are created in this research are used as explorations of compositional ideas and how these relate to emerging computational methods in architecture. The insight into the development of these experiments also allows for refinement and calibration of further experiments (Burry, 2011).These projects additionally hold aspirations for developing techniques applicable outside of singular design instances, pointing to potentials for developing comparable design practice methodology in a broader industry setting.

3.1. Design Research Methodology

Drawing on aims and background research, the methodology attempts to produce a framework for design knowing through design investigation process. This approach involves a series of project-based design investigations in conjunction with literature and precedent review. The projects are coupled with exegetical reflection and critical evaluation. The project-based approach is distinguished from traditional thesis model, which typically begins with a hypothesis based on well-established epistemological conventions and proceeds to examine it by presenting relevant empirical or logical evidence. Since this thesis deals with application of emerging technologies to design practice and existing architectural theory of group form, other additional modes of research incorporating design processes are used which will be discussed here.

Bryan Lawson presents a map of the design process that attempts to approximate it as a procedural sequence (Lawson, 1990, pp. 22-36). Although there are many variations proposed and even according to the author, none of them accurately capture the intricacies of this process, a three-part

[5] The Strategy of reflection-in-action is described in detail by Davis Shon who explains the synthesis of new design knowledge through reflective and critical engagement in practice (Schon, 1983).

flow diagram with varying component definitions and connective topology re-emerges and seems to be an accurate first approximation for a methodological basis of analysis and reflection:

- Analysis is an exploration of relationships, patterns and classification of objectives. Also referred to as 'generator' or the process of formulation of assumptions about a design problem.
- Synthesis is an attempt to create a response to the problem and generate possible solutions
- Appraisal is a critical evaluation of suggested solutions against objectives identified in the analysis stage.

Lawson also draws distinction between behaviour of individuals trained in scientific disciplines and design; through a series of experiments constructed specifically to investigate approaches to design problems by designers and non-designers, he concludes that scientists problem-solve by attempting to understand the rules by maximising information about the problem , whereas designers specifically select information which approximates optimal constituents of a solution (Ibid., 31-32). Presumably, design requires a specific approach for tackling indeterminate problems by emphasising synthesis over analysis. This is partly the cause of complexity of the above-mentioned process map, which assumes a certain order. Rittel and Webber have identified 'wicked' problems originally used in the context of social planning but have applicability in this context (Rittel and Webber, 1973). These are described as ill-defined, ill-structured and fundamentally different problems to those set by scientists and mathematicians who systematically strive to either provide proofs (in case of logic and mathematics) or theories supported by empirical evidence (in case of natural sciences). Apart from the concept of 'wickedness' which refers to resistance to solution which sets this type of problems apart- in the context of design, this epistemological difference has been noted prior:

The scientific method is a pattern of problem-solving behaviour employed in finding the nature of what exists, whereas the design method is a pattern of behaviour employed in inventing things of value, which do not yet exist. Science is analytic; design is constructive. (Gregory, 1966)

Unlike a traditional theoretical research approach, project-based research allows one to directly engage with design process, materials and technologies of design production. The research questions are addressed in this mode through a range of digital and physical conceptual prototypes. This thesis assumes that the real challenge of production of group form is not in conceiving it but rather, in realising it through addressing various 'wicked problems' in relation to scale, composition, technology and process. Thus, the hypothesis of group form as primarily a compositional theory of architecture, which is tested with physical design experiments and project investigations.

This raises a question of what separates an act of design from design as a research programme. Although this is still an ongoing area of discourse, broadly speaking: design practice only enters into domain of research upon critical reflection and methodological approach. Nigel Cross puts this design-centric epistemology in the following way:

"The whole point of doing research is to extract reliable knowledge from either the natural or artificial world, and to make that knowledge available to others in re-usable form. This does not mean that works of design practice wholly excluded from design research, but it does mean that, to qualify as research, there must be reflection by the practitioner on the work, and the communication of some re-usable results from that reflection" (Cross, 2006, p. 102).

This process of critical introspection distinguishes a researcher from a practitioner involved in a non-research activity. Through immersion in material culture and design process, designers create new knowledge through critical reflection and evaluation. This process allows a formation of a design theory model that could potentially aid both designers and design researchers in understanding methodology and decision making in this specific area of group form enquiry.

Figure (insert figure ref here) illustrates the research methodology as an iterative and evolving process based around project-based design investigations in a continuous and reflective process. Each investigation is conducted with

Mark Burry defines three 'scripting cultures' or ways in which designers tend to utilise computational design tools. They are broadly classified into scripting for productivity, experimentation by scripting towards a path to an answer and scripting for 'a voyage of discovery' (Burry, 2011).

In core chapters of the exegesis, design investigations are contrasted with theoretical framework of group form in order to test its technical feasibility by using proposed design methodology and embedded materials and systems at tectonic scales. The key objective is to capture the imagination and project a potential future architecture based on modified concepts of group form for future research. Rather than obtaining quantitative feedback using a more analytical approach, the outcomes of projects are evaluated through qualitative feedback. The feedback process acts as a generative iterative development strategy with produced results used to interrogate original ideas and test the group form hypothesis by developing a new set of inquiries through evaluation and reflection of various aspects discussed.

3.2. Workflow Hierarchy

In order to systematically describe design workflow and depict a systematic overview of processes and methods used on each individual prototype, I draw on the PhD research of John Everett and Nicholas Williams. Everett's research reviews problems in construction industry and the potential for automation technology for addressing issues of productivity, safety, and skilled labour. Of interest to this research is his classification of elements of construction field work which he describes as Levels of Detail (Everett, 1991, p. 59). Dissatisfied with the interchangeable use of construction process terminology, he systematically constructs a formalised hierarchical relationship between them. William's thesis develops this framework into creative practice and introduces the idea of 'levels of detail' in creative research dealing with issues of technological automation and its modularity. These levels could be described as hierarchies of processes- from simple tasks that could be automated to more advanced procedures requiring navigation of multiple disciplines, conflictual design criteria and complex, non-linear workflows. Original schema proposes nine separate categories beginning with a general 'Industry' described primarily as a part of the broader economic output of a given society and going down to the 'Elemental Motion' describing very basic tasks of human movements such as shovelling and walking with load. Of relevance are five levels of detail reimagined by Williams in specific application to architectural production spanning from *Design System (Project)* to *Elemental Motion* (Williams, 2017, pp. 50-54). These assist in understanding workflow at each level or scale and its relationship to materials, digital tools and design computation.

This thesis uses similar framing to William's level of detail, but makes a few necessary adjustments and clarifications. A general assumption is made here that any of these parts can theoretically be automated but only some parts are due to expediency, costs and available (and/or currently understood) technologies. The boundaries between these categories are difficult to describe as the categories themselves are defined by sets of examples and their hierarchical relationship to the neighbouring categories. In Williams taxonomy it is, for instance, unclear what is the essential characteristic of Basic Task and what exactly separates it from Elemental Function (Motion in Evertett). It is described as elements of work with multiple inputs that combine multiple Elemental Functions for more 'significant outcomes'. Some examples provided are that of calculating a geodesic curve (from computation) and applying glue to a surface (a physical task). Compare these to some examples of Elemental Function (process with fewer inputs): offsetting a curve (from computation) and opening a glue bottle (a physical task). Everett makes a distinction between Task and higher complexity categories on the assumption that this work can be performed by a single robot. This, however, assumes that a single-robot task complexity is fixed which depends on the complexity of the

machine itself. If we consider application of glue to a surface and opening a glue bottle as separate categories, a natural response arises as to how objectively compare unknown tasks to determine their classification? The separation of task complexity is intuitive to us: constructing a house is clearly a more complex task than opening a glue bottle. However, when you introduce a gradually escalating complexity categories, the boundaries between various tasks become harder to pinpoint without some analytical tool. Such analysis is beyond the scope of this thesis- these description categories will be applied here with some discretion and at the conclusion of having utilised this taxonomy, some suggestions for improvement of further classification schemes will be made. The following categories are process descriptions based on Everett and Williams.

3.2.1. Project

Everett's description encapsulates a building type under this category, which is a singular event with a definitive starting and stopping points as well as unique, interrelated activities. Williams' transitions this to a more abstract category of a 'Design System' that represents a complete, self-contained, single generation of an architectural prototype. Everett's terminology is retained for its ability to succinctly describe architectural building projects of scales and complexity transcending prototypes or clusters of building sub-systems and in architectural context could be understood as a system of procedures generating complete, habitable building artefacts. Projects presented in this thesis cannot be described using this formulation even though they begin to accumulate several 'division' sub-categories into works that are 'complete' but only in some more abstract, compositional sense.

3.2.2. Division

Major subdivisions of a project process, classified by functionality, the nature of assembly and assigned according to the expertise and project roles of people involved. It is, essentially, a way of organising processes specific to a distinct aspect of the project. Projects in this thesis are composed of these divisions to a limited extent. Examples include form finding, fabrication design, electrical design, interaction design, structural design, architectural design.

3.2.3. Activity

Describes a specific item of work that breaks up Division into multiple procedures. Williams defines this category as a specific preparation or fabrication methodology such as defining a robot toolpath or calculating structural deflections. Everett describes Activity as a procedure that results in a recognisable constructed element. This thesis uses William's definition and approaches Activity as a collection of Tasks that can be described as completing a recognisable unit of work that can either be virtual or physical. An activity could be producing specific connectors or building components, alternatively it could be producing digital models and/or analysis of said parts.

3.2.4. Task (Basic Task)

Task can generally be described with a simple procedure and a series of simple inputs. It does not require significant prior knowledge or expertise. As Everett points out, this is the level at which automation becomes increasingly useful since the problems occurring here are much more tractable in terms computation.

3.2.5. Basic Method (Elemental Motion)

A unit of work that can be described as a specific basic function (usually performing a certain geometric operation constrained by a limited number of input types and effects that it generates). Digitally, this could be a simple method function in code that, for example, discretises NURBS surfaces, transforms objects or calculates a particular relationship between geometry (e.g. linear algebra function). A physical counterpart could be drilling a hole, actuating a kinetic motor or a sensor.

A level of detail beyond Project is Sector, which represents different types of constructed facilities. In Everett's description these facilities range between different general building typologies (residential, public works etc.) to heavy construction (highways and street) as well as infrastructural projects including electrical, water and special trade (Everett, 1991, p. 61). This generalises to complete procedures involved in constructions of clusters of buildings or quasi-buildings. This process begins to suggest urban scale intervention processes that would also include all built matter that connects and supports buildings in these clusters. The processes at this scale were of primary concern to Group

Form concept as articulated by Maki: *"Our problem is this: Do we have in urban design an adequate spatial language (an appropriate master form) with which to create and organise space within the master program?"* (Maki, 1964, p. 4). There is, however, little description of how this program can be effective at all scales of production despite introducing it as a bottom-up process occurring naturally in clusters of traditional villages and settlements that emerge from small, architectural elements. *Figure 3* compares the scale domain of this research in relation to Everett, Maki and Williams. In terms of scale of detail, it is much closer to Williams due to the focus on relatively small-scale material prototypes that explore digital tools in contemporary architectural production. However, the focus of this research is on reconsidering Group Form approach from the point of view of designing behaviours rather than form, and in this sense, Everett's level of detail framework and William's reformulation provide a good foundation for understanding process as modular system of task components both manually executed and automated. Because the research looks at scales other than Sector and Project, it allows for a greater range of levels of detail as well as a more immediate access to material design processes. While utilising this workflow hierarchy model, this thesis attempts to expand the scope of the research programme of Group Form as well as reconceptualising it as a multi-scalar network of modular and unique systems of production and operation.

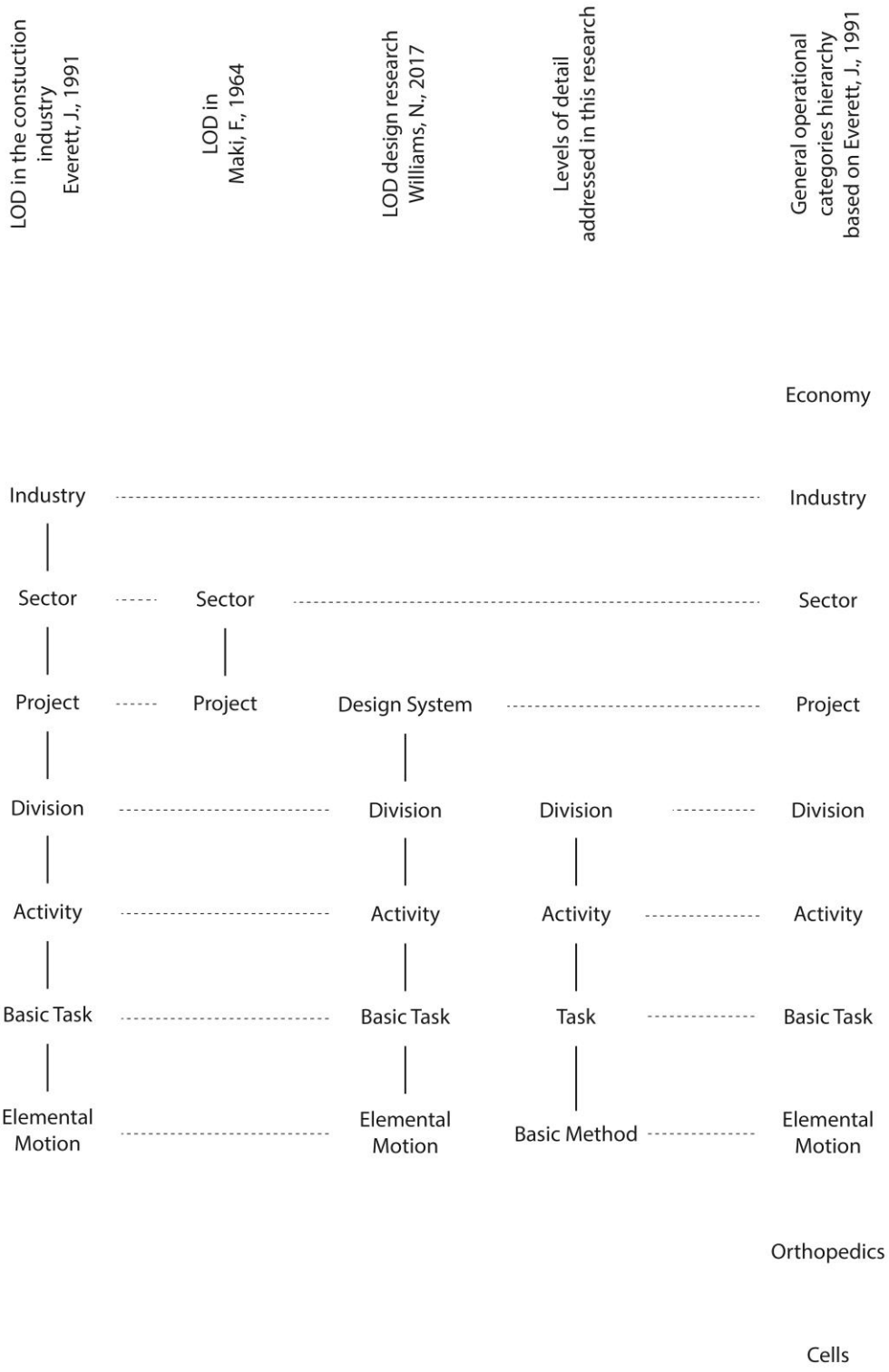


Figure 3: The scales of detail for design systems compared to Everett, Maki and Williams.

3.1. Design Investigations

In the background chapters, I described various parallel directions in contemporary architectural discourse that in one way or another connect to the ideas of group form. These are regularly revisited as key themes throughout the project chapters and further elucidate design ambitions that drive these explorations. They include application of digital modelling and scripting to 'system design' and digital fabrication and its effects on the concept of part-to-whole relationships in design process, differentiation at multiple scales and exploration of relationships between multiple interdependent design drivers.

To connect these themes to specific projects and interrogate systematically different potentials of group form, I set up three forms of research drivers in order to elucidate per-project hypothesis (testable design driver) and key questions around interdependent relationships of geometry, fabrication and performance. These are shown in *Table 1.*

Structural and electronic systems of each project are further constrained by several other factors. Each project is a public installation in specific public space context that is displayed for a duration of several days and is asked to interact with the public. Another factor is primary affordance set within the constraints of a real design project with sets of budgets and deadlines that made them comparable to proto-architectural commissions. Another determinant is the collaborative aspect that varies from an interdisciplinary collaboration to a collaboration across multiple disciplines. Finally, the production of each prototype involves the use of specific fabrication technologies that helps to explicate challenges and opportunities of designing for specific machines at all stages of the project.

Table 1: Key hypothesis and questions formulated for each project:

Design Investigation	Hypothesis on form and performance	Key Question	Design Approaches
Cellular Form	Cellular collective form can be translated from virtual aggregating simulation into a material prototype	What are the challenges in creating aggregative group form (bottom-up) with a grafted system?	• Recursive scripting • 3D printing (Fused deposition modelling) • Simple interactive prototyping
Mediated Form	Collective form can connect and mediate context beyond its immediate physical presence	How can physical forces and locative technologies influence design parameters for group form?	• Site analysis and visualisation through scripting and simulation • Construction of complex interactive system

			• 3D printing (Fused deposition modelling), • 3 Axis CNC cutter
Kinetic Form	Collective form can be composed of kinetically-responsive elements	What are the effects on scalability and top-down control of form in group form with kinetic-responsive parts?	• 3D printing (Fused deposition modelling • Interactive Arduino Prototyping with kinetic elements

The core methodology followed in this thesis is thus a design-type, which focuses on construction of physical prototypes to test, experiment and expand on the operational principles of design of group form. From a theoretical standpoint, this thesis aims explore design approached to group form with contemporary materials and technologies of objects and systems that can encompass multiple scales. To do this, the operational categories have to be treated more abstractly and the design and construction of prototypes is aimed to give an insight into some of the challenges and potentials presented when trying to scale group form into larger assemblies. Additionally, all three projects were displayed publicly during various local and international events to conduct design research within public space and within the constraints of a real design project with sets of budgets and deadlines that made them comparable to proto-architectural commissions. This allowed for examination of theoretical concepts around technology, group form and interaction and at the same time test which methods and techniques are most fruitful in further development of the concept and its potential applications.

3.2. Diagram Conventions

In project chapters that follow specific diagrams are used to describe modular workflow. This is done retrospectively in order to elucidate design methodology and to highlight the bottom-up and top-down forces active in design research (Williams, 2017, p. 53). These diagrams are simplified maps of process, they represent abstractions of complex manual tasks and automated scripts. They do not show difference between Basic Methods and Tasks (shown as blocks) as this difference is difficult to ascertain purely from category descriptions provided in the literature. These categories have been collapsed into parts contained by larger groupings of Activities and Divisions.

In order to create a regular notation which is consistently applied across the project chapters, diagrams obey a number of conventions:

- Workflow components are represented by rectangles (*Figure 4*) of various scales that correspond to their place in processes hierarchy and indicate its level of detail such as design sub-system, Task or Method as described above.
- Workflow components are connected by arrows indicating design flow direction. Some may have a backward direction while other might have an 'else' tag indicating a branch from an 'if' component which will always have another primary arrow in case the statement is true.
- Workflow components and corresponding to them arrows have differentiated line types. These indicate the level of modularity so that the continuous lines represent full modularity, dashed lines represent semi-modularity and dotted lines represent uniqueness.
- To represent workflows operating at various scales, workflows components are nested inside one another.
- Filled in components indicate part of the process that is either creatively complex and involved many discretionary choices or that it is very labour intensive and involved complex assembly tasks. In all case, these processes are manual.

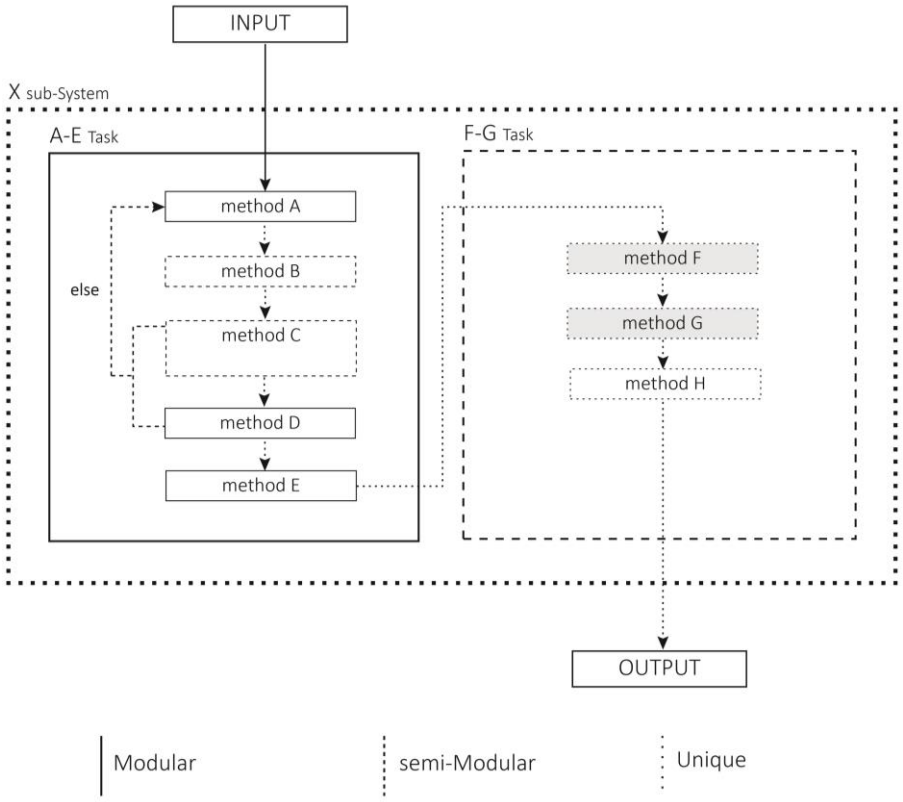


Figure 4: Typical diagram structure showing an Activity composed of Tasks that are themselves made out of method blocks. Modules with lesser dash frequency (dotted<dashed<solid) indicates a decreasing degree of modularity.

Design Investigation1:
Cellular Form

4. Design Investigation 1: Cellular Form

This inaugural project[6] was the first step in understanding the implications of combining computational design, aggregative geometry and system integration into architecture. It comprises of multiple relatively simple design processes that come together to form a deeper understanding of some of the problems and potentials of group form approach. The initial prototype formulates a bottom-up design methodology tracking part-to-whole relationships and their arrangement discussed in Background that had to satisfy the following criteria:

1. The prototype is based on a component that has the ability to aggregate into larger wholes.
2. The prototype features a behavioural property that requires being scalable and modifiable in relation to component geometry.
3. Assembly has to be reconfigurable to allow for modifications and replacements.

This chapter is divided into four parts covering all aspects of the design, fabrication and installation processes. *Investigating Scripted Aggregations* will look at the initial design methodology, which experimented with aggregative scripts and physical modelling. *Digital Fabrication and Material Testing* covers full-scale development and fabrication of components. *Reactive System* covers interactive technology used in the prototype as well as the challenges of integrating it with non-linearly tessellating components. *Discussion* will cover which strategies proved successful and which needed further investigation and refinement. It also develops the framework which encompasses the following project and *Summary* will conclude the chapter by teasing out conceptual implications of the prototype and the direction of future research.

4.1. Investigating Scripted Aggregations

The initial process took inspiration from aggregation as a system-type approach. Instead of pre-defining geometry and physical characteristics of the prototype, a diagram of relationships for a variety of potential configurations was developed. This took form of a simple script which took a simple geometry element (b-rep or boundary representation) as input and put it through a recursive process. The algorithm imitated a random walk using this initial input and undergoing the following procedure. First, it picked a random face of the b-rep, copying and mirroring the shape around a plane derived from a normal of the chosen surface. Then, using Rhino Geometry library, we check for

[6] Working title- reference to the appendix

any b-rep intersections between current shape and a list of all previous shapes stored in a separate list. If none are found, the b-rep gets added to that list. This process is described in detail in *Figure 5.* The primary loop (enclosed by dashed lines) indicates the primary mirroring transformation followed by a selection process towards the bottom that runs an intersection test to determine if the current shape clashes with previous shapes. These are, in turn scaled to a small factor (0.99) to prevent them from detecting intersections between intended direct neighbours. Thus, the procedure keeps track of two lists- *list b* is the collection of all non-intersecting scaled b-reps and *list a* is the final list of all the non-intersecting complete b-reps.

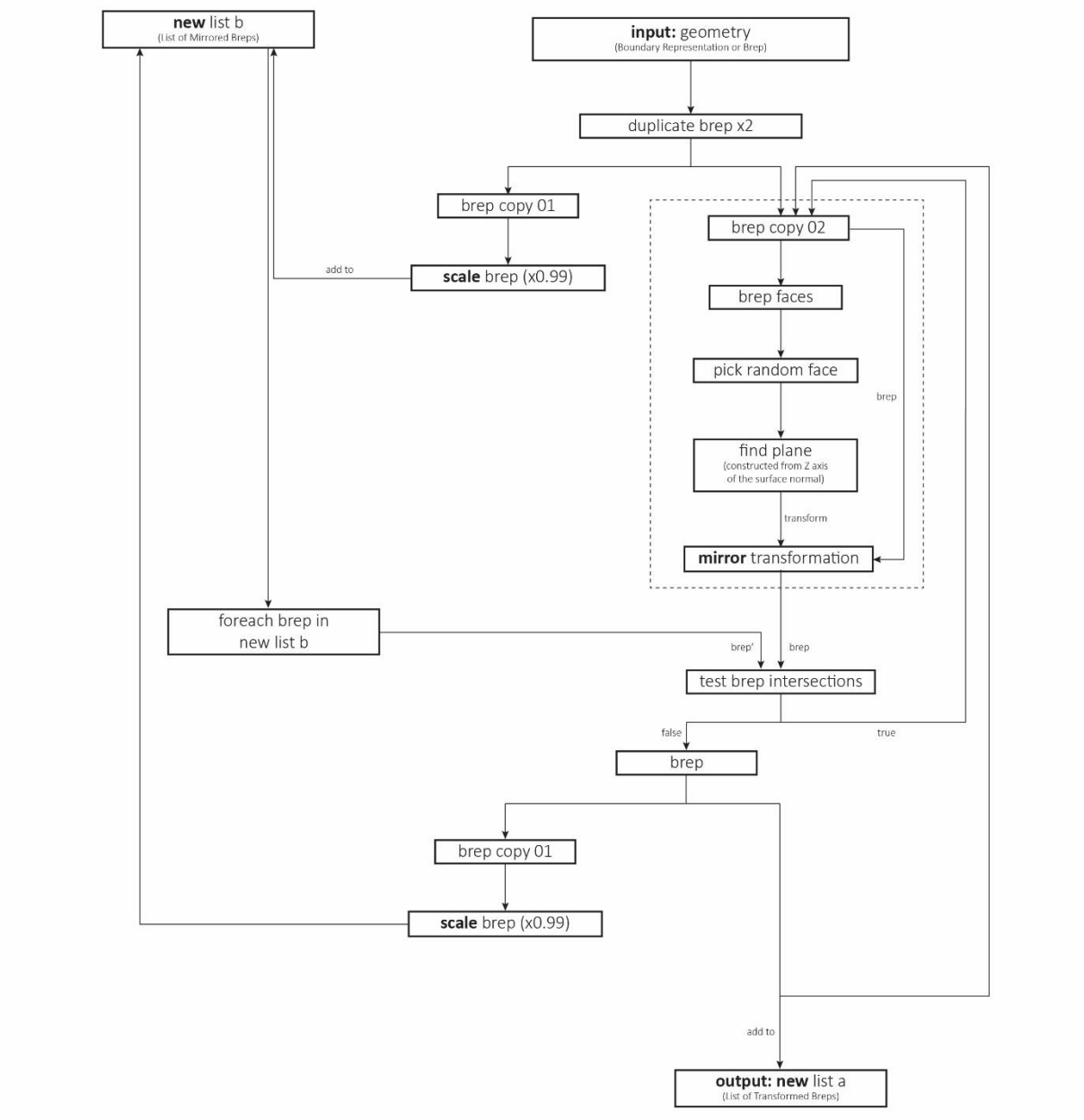


Figure 5: flow chart of random walk algorithm.

Face selection function is randomised with a standard .NET function with a seed supplied from system millisecond counter. This was done to avoid repeating patterns in aggregation, as with pseudo-random generation, after enough iterations, the rhythm and pattern becomes noticeably repetitive. A consequence of this is that it also behaves non-deterministically, producing different outcomes with each new iteration.

Because input geometry is open, the procedure allows for a wide range of experimentation with varying starting geometries (Figure 5). All starting shapes in these examples are closed, non-self-intersecting, symmetrical and composed of all flat surfaces and derived from platonic shapes (the last condition is optional). In some cases, the form will get self-locked where the algorithm gets stuck in a mirroring loop preventing b-rep from being mirrored and continuously checking adjacent positions. For these cases, an upper limit on the number of allowed random search selections is set to a low number (10) for the aggregation to exit when it encounters self-locking. This set up creates a constraint on the upper limit of parts during an average run which appears to be determined by the concavity and the number of faces of the input form of the random walk algorithm. The cube-based configuration (6 faces) appeared to be the most stable, reaching the upper limit of 200 parts consistently. The tetrahedron configuration (4 faces) was least stable, only seldom reaching the upper limit, and often locking out half way at around 100. Other configurations with larger face count per part such as dodecahedron (12 faces) and truncated tetrahedron (8 faces) yielded more unpredictable numbers of parts per run.

For physical prototyping, truncated tetrahedron part was selected. It is similar to a standard tetrahedral model, but by truncating it such that the big faces are regular hexagons, the overall enveloped space is increased, creating a longer chain of parts and more complex internal cavities. Another interesting characteristic of this form is that it is potentially not facet-transitive: the way in which tetrahedrons array in most cases creating gaps- for instance groups of five (create non-facet-transitive rings) and groups of twenty (forming non-facet-transitive spherical shapes) leaves gaps in a non-alternating arrangement. [mention aperiodic tessellation].

The prototypes were 3D printed (edge length) to test the possibility of creating facet-transitive forms by exploiting inexactness and physical tolerances of materials and linkages between parts. *Figure 7* shows these parts joined into links, taking advantage of material tolerances in the physical prototype. It proves, at least at small scales, that the incompleteness of the digital model could be breached by the physicality of the elements produces in the real world. The components produced facet-transitive rings as well as spherical configurations without any perceptible misalignment. The physical prototyping also showed that using a relatively simple process, the algorithm created on the

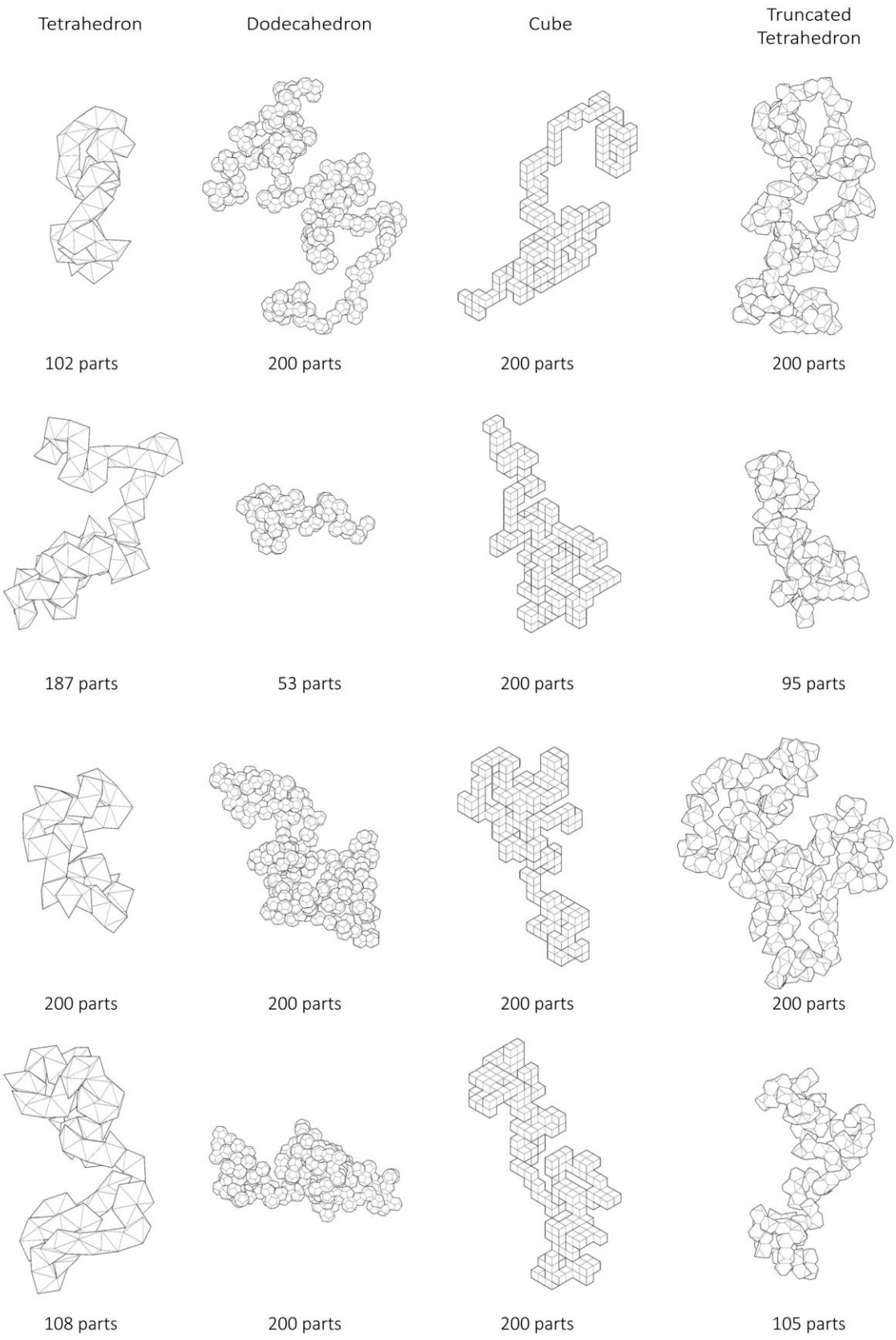


Figure 6: Random walk aggregation script output.

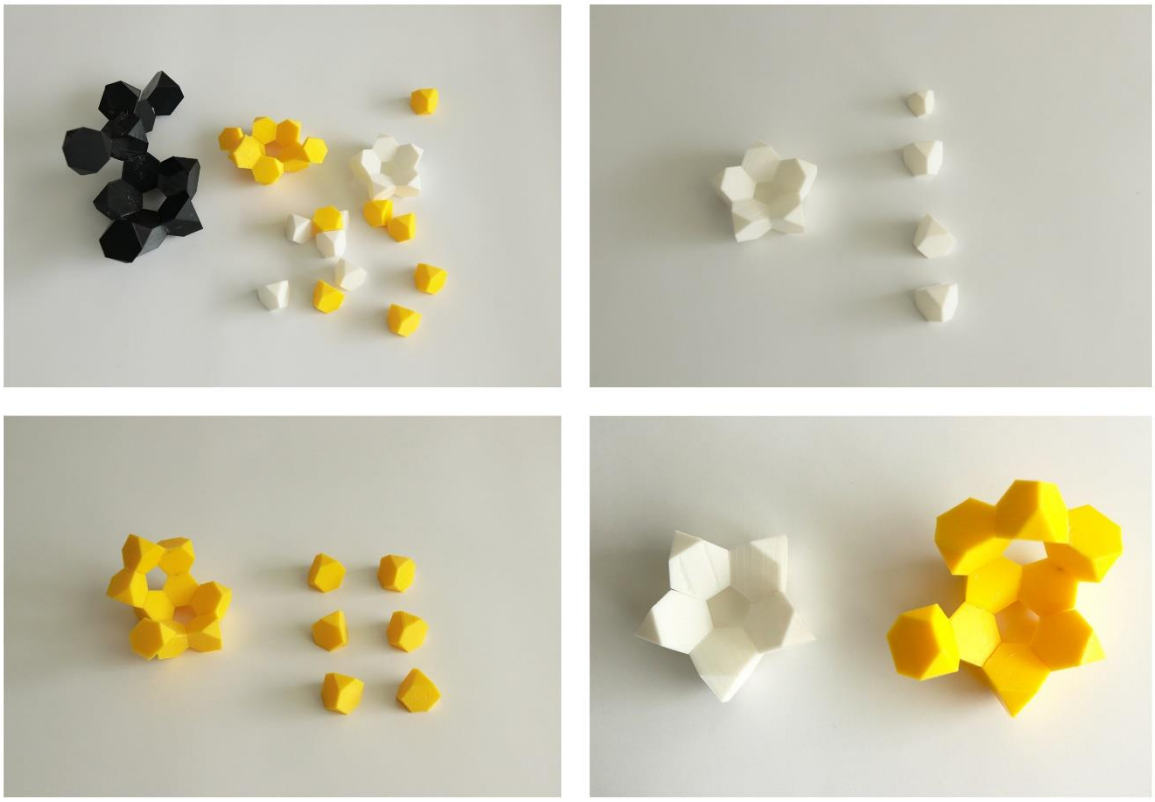


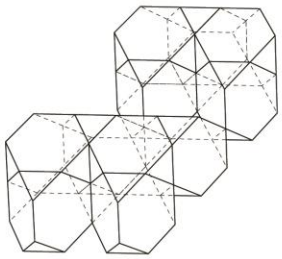
Figure 7: 3d printed tests.

computer could easily be reproduced using physical design process with the added advantage of bridging the gap between parts with physical tolerances. To test whether this held true for larger assemblies, a large-scale prototype (edge length) was built which is described in the following chapters.

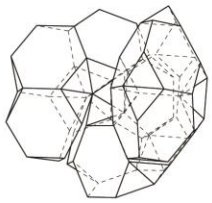
The closing of the pseudo- spherical form was also explored with the digital model. It was clear that the tetrahedrons were aggregating into recognisable patterns even though the misalignments were present and compounded the longer the chain of aggregated parts became. An inverse process was also tested: by starting with the whole (enclosed spherical configuration) and subdividing it into smaller parts, we were able to achieve rings and spheres that were consistently tessellating. *Figure 7* visually shows various iterations involving aggregating components and the inverse process of starting from a regular dodecahedron.

One of the known space filling configurations is truncated honeycomb (with Triakis truncated tetrahedral honeycomb variation having the property of completely filling up space) shown in the top right (***Error! Reference source not found.****.1).* If we, instead, join tetrahedrons in such a way that h

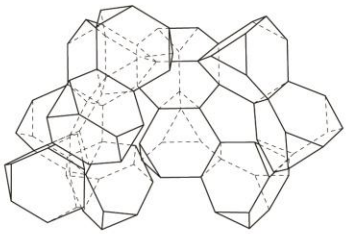
exagonal faces touch while smaller triangular faces align, the configuration begins to close into spheres without being able to form complete forms, with gaps growing ever larger as the aggregation continues (*Figure 8.2).* The same property occurs when we aggregate by joining small triangular faces (*Figure 8.3).* However, if instead of aggregating components, we begin by constructing a singular spherical configuration from a truncated dodecahedron (*Figure 8.5)* and use its triangular faces as interior faces of tessellating tetrahedral parts, while using a truncated icosahedron (*Figure 8.6)* and its hexagonal faces as cross-sections of the tetrahedral parts, we can build a parametric conic extrusion (*Figure 8.7)* which serves as the basis for a complete spherical structure made of truncated tetrahedral parts (***Error! Reference source not found.****.8).* Although, the advantage is that the whole is c ompletely enclosed, the disadvantage is that the parts become irregular, creating more complex fabrication problems as the edges and faces have different sizes and shapes and the way they have to connect is very specific. Another disadvantage is that the spherical assembly itself cannot be tessellated into larger enclosed configurations using truncated tetrahedron, even though many tessellating configurations exist- a few examples are a bitruncated icosahedral honeycomb for truncated dodecahedrons and bitruncated order-5 dodecahedral honeycomb for truncated icosahedrons (references needed). These rely on fairly complex topologies and multiple cell types that significantly problematized intended assembly fabrication process. For these reasons, a more simplified approach was used for larger scale fabrication which kept the simplicity of spherical aggregation of truncated tetrahedrons whilst maintaining facet-transitivity through material elasticity and physical tolerances.



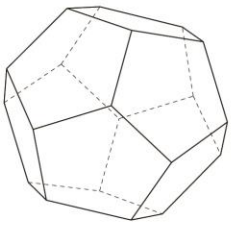
1. Truncated
tetrahedron honeycomb



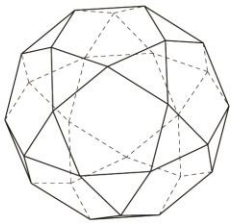
2. Spherical truncated
tetrahedron honeycomb ver.1



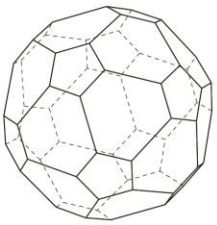
3. Spherical truncated
tetrahedron honeycomb ver.2



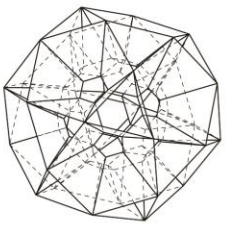
4. Regular dodecahedron
(approximation of the spherical honeycomb
aggregated form)



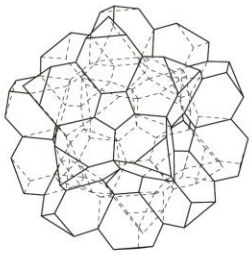
5. Truncated dodecahedron



6. Truncated isosahedron



7. Conic extrusion from
truncated icosahedron (interior)
to truncated dodecahedron (middle section of
truncated icosahedral parts)



8. Morphed truncated tetrahedrons
constructed on truncated dodecahedron

Figure 8: Part-to-whole relationships studies.

4.2. Digital Fabrication and Material Testing

Original intent for large scale fabricated prototype included a sample of the output from the algorithm described in the previous chapter, modified to be focused around three complete spherical assemblies connected and reinforced with additional parts. The prototype was meant to answer a number of theoretical questions that have to do with a transition of a digital/speculative model into physical/tangible space. One of these is whether material flexibility and tolerances of the real object would allow negation of non-facet-transitive nature of the chosen configuration. From small-scale 3D prints it became clear that this could be achieved given enough flexibility and tolerance allowance in the connections. Another question arose out of aggregating spherical forms derived from dodecahedron geometry into larger assemblies on the computer- how far the physicality of the real object could potentially negate geometric certainty of the digital geometric 'incompleteness' and whether it was possible to push the facet transitivity up a scale without it being the case in the simulated environment.

Figure 9: Rendering of proposed large-scale structure.

Another feature of a larger prototype design is the integration of secondary system into the assembly- this is the first test of incorporating a simple responsive scenario into the logic of the geometric composition. It takes advantage of the linear aggregation of components in the script, treating them as modular, re-buildable pieces and at the same time as a linked chain of reactive components. In this prototype, we used custom assembled LED strips that can be linked interchangeably. The strips are built into the cells to manifest and connect interactive presence of the object with the process that originally gave rise to it. The interactivity and its assembly is described in more detail in the following chapter.

From the intent of fabrication and assembly, components were made to be solid objects enveloped by a translucent material while at the same time maintaining modularity and facet-transitivity at larger, physical scales. Support cells were designed as skeletal constructions, manually inserted at points deemed necessary for the structural integrity of the physical object into the digital model (*Figure 9*). This representation used the top-down approach of spherical part construction described in the previous chapter where the three main clusters of parts are made from regular truncated platonic forms. For the construction, regular component representation was used instead. This was done partly for the purposes of visualisation (to remove incongruent non-facet-transitive sections from the model) and to give the ability to place connective cells whilst maximising the accuracy of local connections and their position relative to all parts of the model.

The construction of the large scale prototype took form of structural and material tests at increasing scales and complexity (*Figure 10*). First the two main components were assembled and tested for efficiency of fabrication and ability to be connected interchangeably with each other and members of the same group. This was key to the modularity of the entire assembly- maintaining structural stability whilst using mechanical joints as points of connectivity. One of the problems that occurred during fabrication was connectivity between multiple translucent parts. Due to their lack of internal structure and the use of friction hinge joints that were keeping the solid faces in place, a configuration involving multiple solid parts proved to be structurally unstable. The solution involved a change in assembly strategy- the focus was placed on constructing complete rings of solid components in stages. For one spherical assembly, this involved construction of three complete solid ring components (with prebuilt connected flow) and additional structural frame pieces. This method of construction sequence proved to be most effective for assembly, transportation and disassembly as it minimised the number of points of connectivity while maintaining portability of pre-assembled parts.

In terms of physical construction, assembling component parts was made relatively simple given the self-same geometry of all component parts. The hollow pieces were fabricated with pre-cut 3mm pine

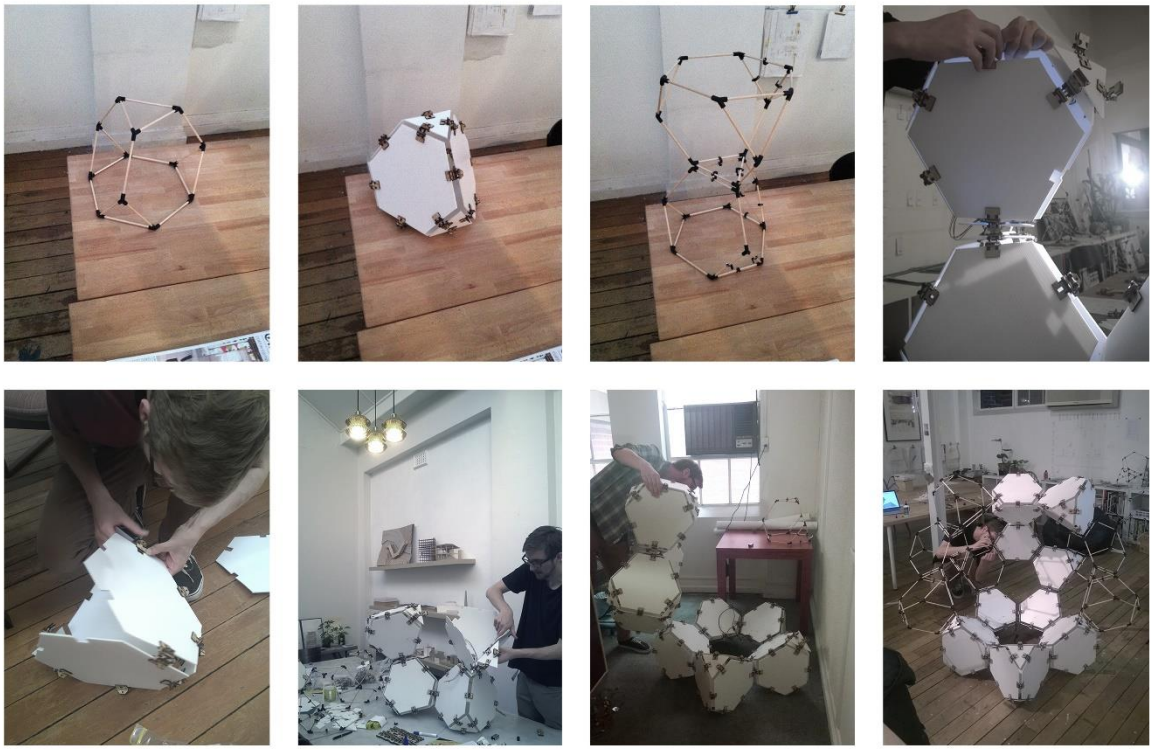


Figure 10: Full-scale prototype testing and construction.

dowels and 3D printed connector joints at vertices. Translucent components were made from multiple materials- faces are made from milled 2mm polycarbonate flute core, joints are a combination of laser cut ply and bolts that act as axis for variable angles between surfaces in the assembly of truncated tetrahedron parts. The process of prototype assembly showed that although the hinge joints were modular and gave structure a certain aesthetic quality through their tectonic intricacy and separation of faces, allowing light to define edges of components- their quantity and individual complexity proved difficult to mass produce. Due to this, we limited the scope of the final assembly to a single form derived from the truncated platonic forms and containing physically facet-transitive parts which, through imposed tension and tolerance correction, caused additional rigidity and stability to the overall structure.

4.3. Reactive System

Metabolic design process also requires designer to consider design of systems- modularity and indeterminacy of performative characteristics of the assembly has to be fluid- the form has to accommodate process in flux (Maki, 1964). This transitory nature of design matter and design of systems requires a modular approach not only of physical components of the assembly but also of the potential function of these components. The part has to be able to perform individually or as part of an ensemble of an indeterminate number of parts and as well as topology. This project introduces a very simple, experimental system based on interactivity in order to explore potentials of complex systems in structures emerging out of simple rules.

At the core of the system is input, received by assembly from its surrounding context, and output which is being communicated back to its surrounding, having been discretised into modular cell units. The input is sound, received by an inbuilt microphone, processed by an Arduino microcontroller installed at the base of the assembly- a component part which was used as a seed for the aggregating algorithm. A simple C++ script converts sound intensity into RGB values for LED configuration of the output (*Figure 11*). The output is composed of discretised water-proof LED components, connected to two crimped wires with a male and a female sockets, built into two hinge connectors on the translucent component. This, in principle, allows different components to be joined together at specific junctions, creating a reconfigurable array of light cells that is consistently 'aware' of its coiled topology.

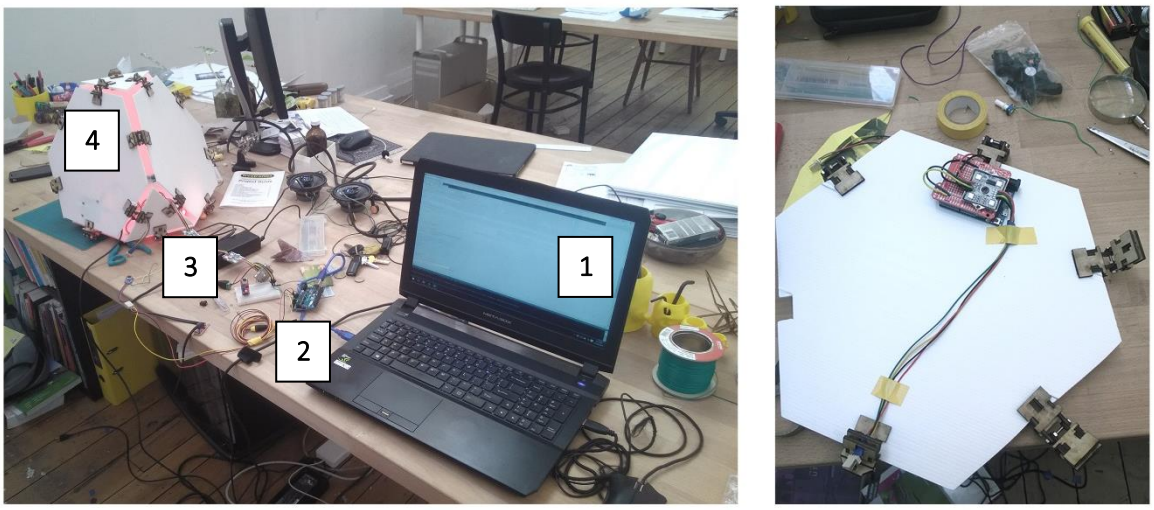


Figure 11: Fabrication of Interactive Systems 1. Laptop running test C script, 2. Arduino Microcontroller used to control the chain 3. The chain of LED components 4. Physical component assembled with connectors and LED.

4.4. Discussion

This project is the first approach in series of conceptual explorations of group form design principles at smaller scales. It embodies a cellular approach of both its formal and systemic characteristics- the process is conceived from a component part that was given a logic of 'growth' mainly based around its geometric characteristics and given further additional system or behavioural properties that are also modular.

This investigation explored group form's operational categories of *Repeat* and *Make Path*. These categories of group form were experimented with by starting with random walk procedure logic creating bottom-up configurations of parts, which were then constructed physically through further design and prototyping. The category of repeat created two types of components- solid parts and frame parts. Solid parts were connected to each other in a linear progression, forming a *Path*. The path was then structurally supported by adding in additional framed elements that complimented it structurally.

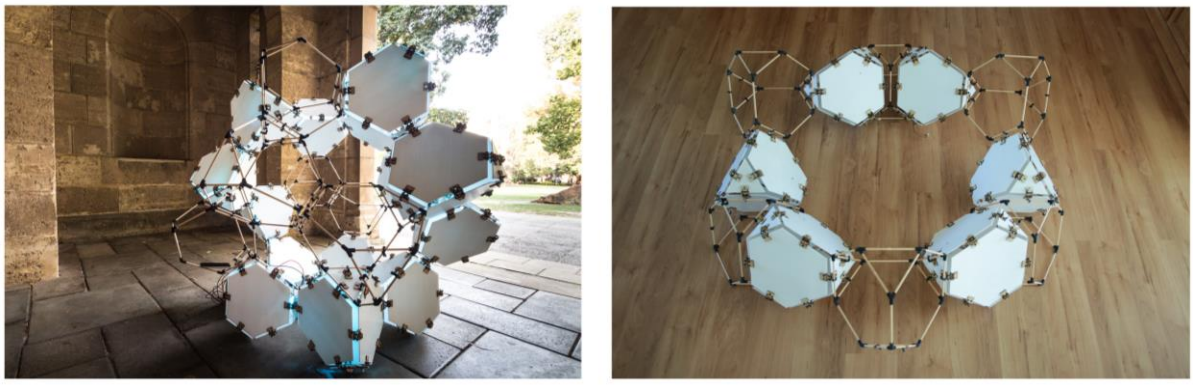
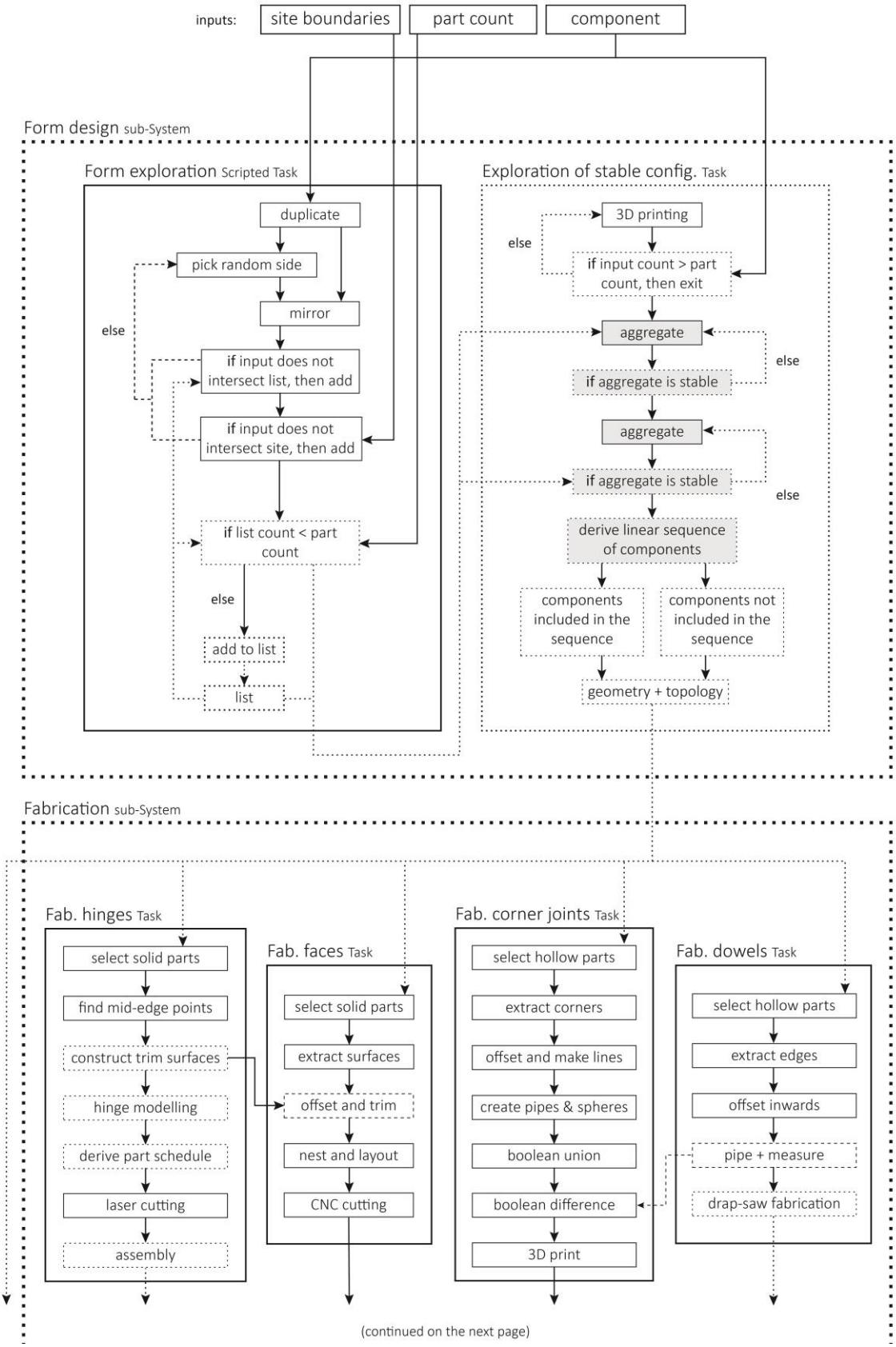


Figure 12: Left- Assembly during the exhibition, Right- re-assembled components

In reality, due to structural constraints mentioned in the previous section, the light connectivity was fixed locally within ring sub-assemblies. Configuration, therefore, was constrained by the location of I/O sockets on individual components and sub-assemblies- an issue which arose out of the necessity to connect LED physically in order to communicate and feed power to the setup. This compromise largely constrained configuration of components to a particular sequence. This compromise could be overcome by increasing the numbers of I/O sockets per component, or compartmentalising systems into discrete cells with wireless connectivity and separating signal and power channels.



(continued on the next page)

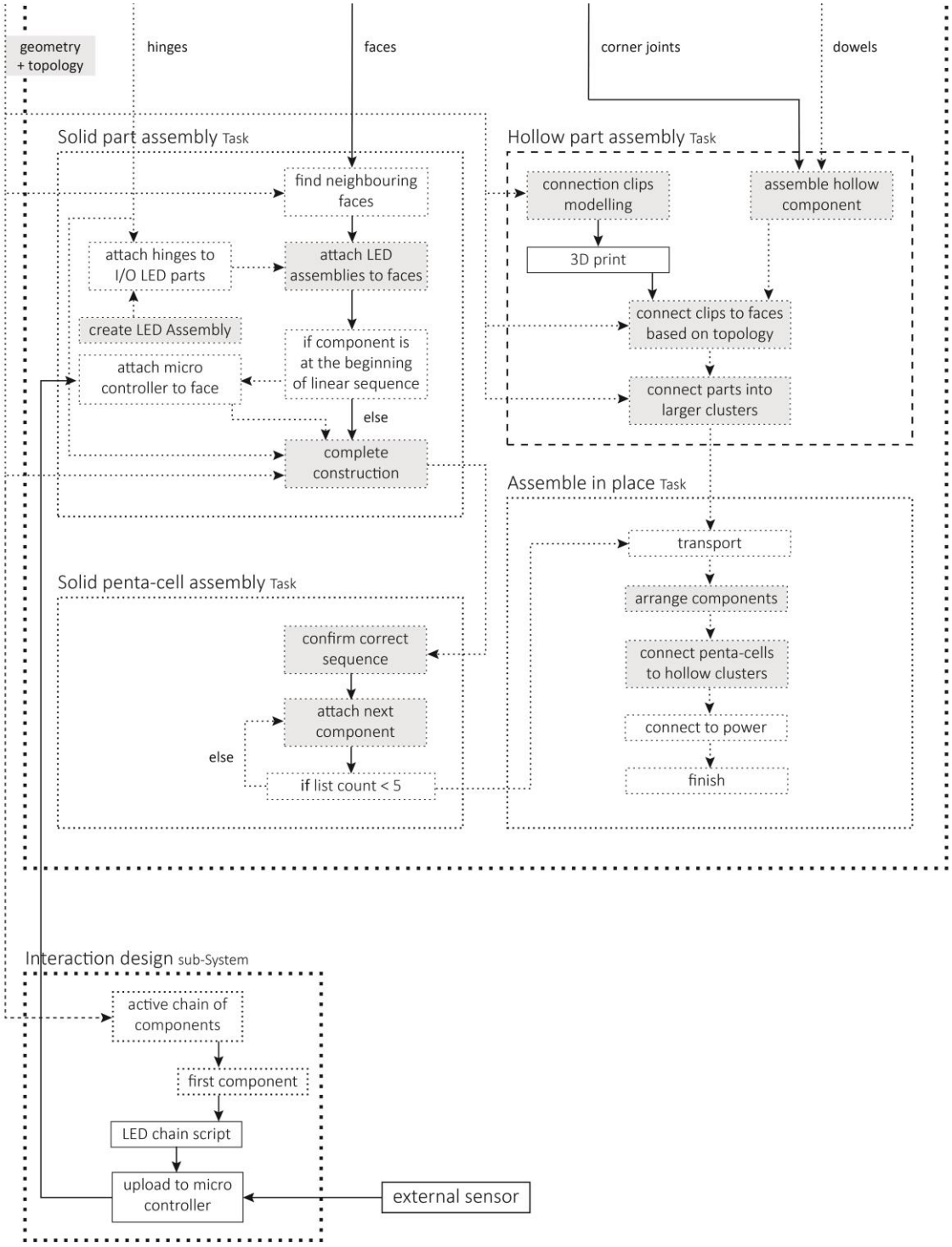
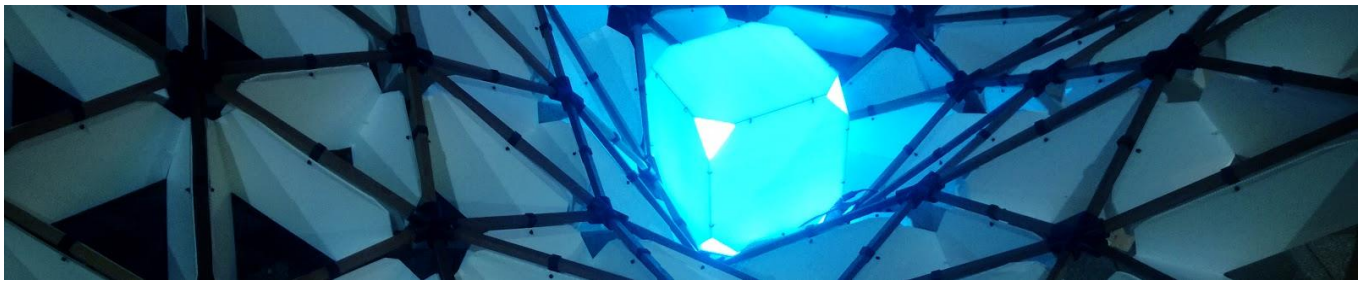


Figure 12: Modular Workflow Diagram for Design Investigation 1

4.5. Summary & Discussion

Overall, the systems experiment showed that with a relatively simple component configuration, one can achieve an integrated structural and discrete-interactive assembly that has the potential to be fully reconfigurable. Interactivity, in this case, served as a placeholder concept that extends to more practical and architectural applications. Responsiveness to internal and external environments, topology configuration and even self-assembly could be potentially explored in component-type aggregations and relate imbedded system logics.

One of the challenges demonstrated in the process of design and making of cellular form is maintaining precise control over the assembly sequence. This is because at early stages of the design, it is difficult to foresee the whole assembly constraints- the aggregation process is difficult to control both practically and conceptually. The communication of parts requires a high degree of pre-planning relating closely to the increasing complexity of the systems in the design. One of the major limitations of the project is its scale- in the context of group form, which concerns itself primarily with urban aggregations; cellular form project was largely disconnected from its location and context.



Design Investigation2:
Mediated Form

5. Design Investigation 2: Mediated Form

The second project explores categories of selection and mediation. Selection could be understood as connection of group form to its context through the use of prominent topographical features or the existing urban forms. Maki gives a number of examples from his urban studies including a Japanese agrarian village expanding out from and following the geometry of the articulated circulation spine of the main road, townscapes of Manarola which features house unit orientations along its natural terrain among many others (Maki, 1964, pp. 43, 46-47). The site is understood as having a set of potential or, more formally, parameters that influence group form. The examples demonstrates formal 'adaptation' to context. This investigation will look at more intangible connection to site that is related to modern distributed communication networks and the possibility of group form to participate in this ethereal and technological context to influence the designed behaviour.

Mediation is understood as connection of parts with a secondary compositional medium and in the urban context it is the linkage between different program through some morphological transition. Maki presents Dutch verandas (stoeps) which create this shift between public and private program and serve as a functionally unifying feature (Ibid., 38). Similarly, Allen describes a moire pattern as an *'*analogous effect of superposition*'* (Allen, 1997) of two regular fields and provides his diagrammatic variation on the concept noting its potential for producing emergent figures. Mayne presents 'variation and overlap' as an urban strategy which take multiple discrete strata and overlaps them, creating new hybrid typologies (Mayne, 2011, p. 43) often through the use of self-similar Boolean subtractions from the continuous surfaces and collections indicating through morphology variable distribution of functional similarity. This design investigation develops a new concept of mediation between primary elements of group form and a separate but overlapping field, which exists around it and produces emerging effects combining the form, the field and live site data to create a combinatory group form.

The project was undertaken by the author in collaboration with Robert Cameron [7] as part of the Public Platform competition organised by Form [8] and for which the authors were shortlisted. The project was exhibited publicly during 2016 in Claremont Quarter, Western Australia.

[7] *thedeadpixelproject* is a design collaboration between Andrei Smolik and Robert Cameron. Robert Cameron is currently a PhD candidate at the University of Western Australia. His research focuses on exploring of ideas related to the intersection of technology, cybernetics, architecture and contemporary culture. See: http://www.thedeadpixelproject.com/about.html

[8] *"FORM is an independent, non-profit cultural organisation that develops and advocates for excellence in creativity and artistic practice in Western Australia".* See:.http://www.form.net.au/our-organisation/about-form/

5.1. Exploration of Information Fields

In order to develop a set of site analysis tools, the context was conceived as information fields- having a secondary 'geography' overlaid onto the physical environment and consisting of invisible fields- forces and hierarchies created from commerce, distribution network and communications on social media. Information derived from internet platforms and geometric analysis was visualised by mapping it onto a hybrid 3D model. The site was digitally modelled using photo-references, topographic data and street maps. Data-scraping programs were developed using Processing framework to gather information about networked presence of actors occupying the site. This information was then overlaid onto the 3D model to create a diagram of spatial flows, accessibility, business locations, demographic information and social media data (Figure 14). The simulation was created by combining spatial analysis and social media data. The latter was used to create 'flags' on the digital model which had assigned to them weights based on their social media traffic. In the digital model, a field overlay was created and used as 'medium' for agent based simulation and visualisation technique (Figure 13). Creation of scripting tools for the project was motivated by the development upon certain aspects of the previous design investigation. One of these, was the lack of site specific constraints on the creation of form. This resulted in geometric parts being multiplied in a sequential manner but having no contextual relation to its surroundings neither formally nor through the process of generation. As Maki points out:

It may be easy for someone to invent a geometric form and call it a group-form because such forms have characteristic of being multiplied in a sequential manner. This is, however, meaningless, unless the form derives from environmental needs. Geometry is only a tool of search for group form. (Maki, 1964, p. 20)

Thus, the method shifted towards developing approaches for analysing the site that could affect and change the group form but also present potentially novel techniques for large scale urban analysis. This called for an examination of interdisciplinary relationship between architectural design and, what Dade Robertson describes as *Architectural User-Interface (AUI)* which develops on the ideas originating in the early 1990's by various researchers including Mark Weiser and Tangible Media Group (Dade-Robertson, 2013). These groups and individuals were interested in a novel kind of interaction with computers- early projects introduced the concept of 'embodied virtuality': the idea that our interactions with computer systems could become part of the designed physical spaces. Today it is sometimes referred to as 'ubiquitous' or 'pervasive computing' and exists as a networked collection of servers, personal computing devices and other electronic components (Wiberg, 2015).

The ideas of pervasive computing, which did not exist during the publication of *Investigations in Collective Form*, potentially modifies its possibilities. Some have suggested that the role of urban designer is diminishing, along with the architectural form due to ubiquitous computing having an influence on our behaviour in favour of distributed networked social space. However, if group-form takes advantage of these distributed information technologies, it has the potential to use it to guide its process-driven growth and evolution. This creates the possibility where even simple systems, coupled with surrounding networked fields could generate complex and embodied effects similar to a modern smart phone or a self-driving car.

The agents represented a movement of simulated actors through space, affected by field distortions in the weighted field. Their original starting positions were determined by on-site count of people moving through key entry points. This technique determined the siting of the project (approximate median space coordinates of the weighted flags) and some geometric form adjustments which will be described later in this section.

Another aspect of design investigation 1 that lead to methodological adjustment is focus on bottom-up strategies that made the top-down compositional approaches less prevalent and somewhat arbitrary. The top-down constraint was the physical limit to materials as they accreted into a larger form. By using physical prototyping which acted as a search mechanism for the most stable configuration, the final assemblage was found. This presented an extremely narrow set of options that were ultimately constrained by the structural design of the parts themselves. Thus, design investigation approach for form finding was reversed and the constraint on the distribution of parts within the form was formulated as primarily top-down. The formal approach in this investigation emerged from computational simulation of forces, specifically looking at digital simulation of catenary networks (Roland, 1970). Within the scope and budget of the project we were unable to fully explore the potentials of real-time kinetics outside of the virtual simulation[9]. Instead, the systems of interaction between context and the artefact were generated through sound and light actuation described in section 5.3: *Reactive System*. The three general principles that were used as inputs for the geometry included:

1) Starting mesh geometry (shape and scale of tessellation)
2) Number and position of anchor points (fixed vertices of the mesh affected by the weighted field).

[9] Virtual simulation of forces on the mesh was achieved with Kangaroo Physics plugin for Grasshopper: http://www.food4rhino.com/app/kangaroo-physics

3) Inverse gravity applied to simulate mesh in compression

During development process, it became clear that a carrier skeletal frame was necessary for structural reasons and the interactive part had to be reduced to one element- the cube core derived from internal anchor points of the Kangaroo simulation. The form of the components evolved from individual sensing/interaction parts to more general amplification cones for sound that were linked to the original triangular mesh subdivisions. The positioning of the anchors was manipulated to add or to subtract the amounts of arches, their height and width and relationship to one another through their framing of context and relationship to the field diagrams.

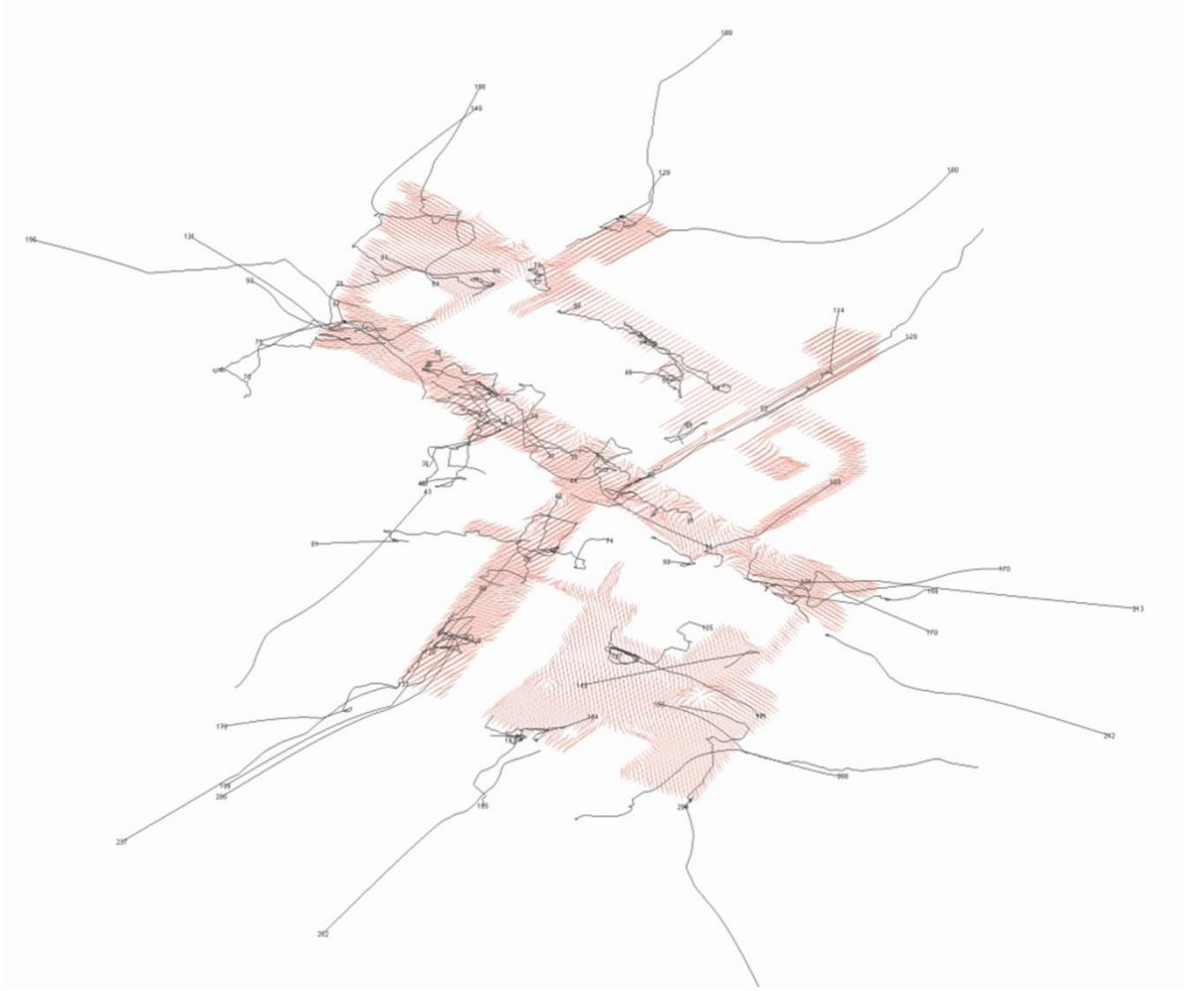


Figure 13: Simulation of swarm agents on a 3D map loaded with collected data. The full video is available here: https://vimeo.com/152675400

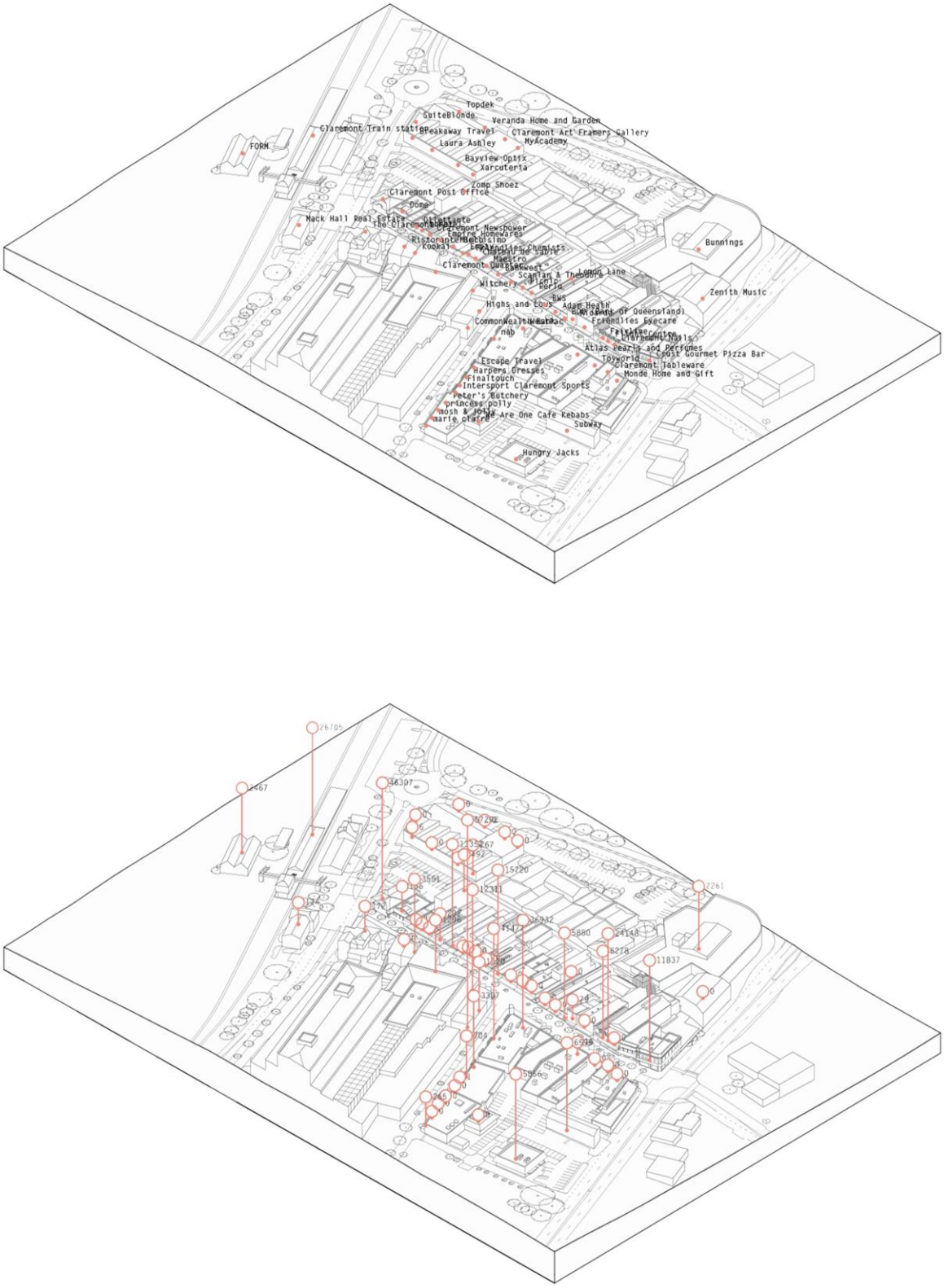


Figure 14: Top map- business names and locations, bottom map- Map of project site with Facebook and Twitter data.

5.2. Digital Fabrication and Material Testing

'Form-finding' in architecture and structural engineering is a method for identifying optimal structural shapes to simulate a specific mechanical behaviour. The reverse-hanging is the oldest and most widely used approach for arches, vaults and shells where physical model, made with elastic cables or surfaces with no rotational stiffness, were subjected to gravity to work out a state in tension (known as 'funicular') and then inverted to deduce a mechanical compression state of the model. [10] With the advent of new digital technologies in the fields of architecture and engineering, it is now possible to use topology of the structural system as the object of the optimisation process itself. [11]

The form was conceived as the process or a force acting on a collection of elements. In this investigation, these elements constituted triangular faces, connected together to form a mesh. The force giving form to the global deformation is a result of the switch in the methodology to prioritise top-down approach. A digitally simulated reverse hanging method was used for top-down form finding. This information became a tool to refine geometry as applied to site with a set of specific constraints that included scale restrictions, availability of material and construction tools, scale of individual component fabrication, orientation and framing of arches and general legibility of form against its context. Although the method itself does not result in a group form, mechanical constraints forming top-down limit on its formation constitutes an important element in considering how bottom-up aggregation could be limited.

Unlike aggregative structures discussed earlier, the form of *Select+Mediate* is gradient-based where individual parts are varying across the span of continuous surface due to the mesh deformation under imposed load. This presented a practical problem of how the form was to be fabricated and constructed. Due to unique part variation, digital fabrication methods had to be developed. The frame of the object was made with 3D printed vertex connectors and Jarrah (Eucalyptus marginate) channels cut to different sizes. The cladding parts were made from corrugated plastic or corriboard panels of different shapes that were cut and labelled using CNC mill[12]. The panels were attached to the frame using 3D printed clips. The structural frame and cladding systems are shown in *Figure*.

Variation of parts and the manufacturing process necessitated development of several scripts that allowed the production and assembly of these parts. The vertex connector script takes into account

[10] This principle was first mentioned in a publication by Robert Hooke: (Hooke, 1676)

[11] Descriptions of contemporary methods for free-form finding and optimisation with complex design criteria can be found in (Sasaki, 2005)

[12] CNC refers to computer numerical control by means of executing pre-programmed sequences of machine-controlled commands. A CNC mill is able to move the spindle (end effector) to various locations and depths on the cutting bed.

topology of connections between edges and using simple trigonometry to minimise material expenditure, creates shapes that are fitted into vertex positions to connect channels. The channels scripts takes the offset input generated by vertex connector script and creates physical target lengths for channels. A knapsack script was also developed to minimise edge material expenditure[13], but during edge cutting it proved as more of a hindrance since the packing could easily be achieved by intuitively selecting edges from a list rather than following an additional list of figures on top of edge length selection and labelling. All scripts involved extensive labelling of unique parts, while the construction necessitated re-building of parts into the shape of the digital model. This also required creation of diagrams or shop drawings to aid the fabrication process and make sure the construction closely follows the arrangement of parts in the digital model (*Figure 15)*.

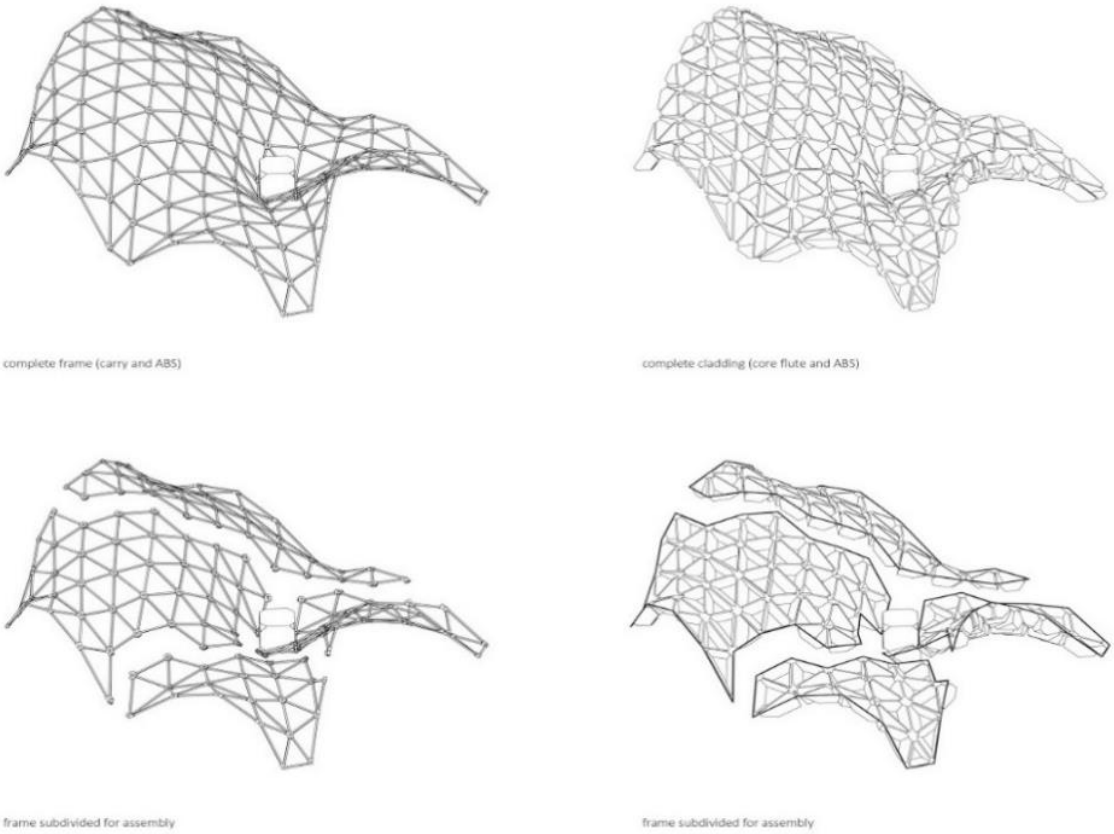


Figure 15: Isometric diagrams showing subdivision of structural system from top left clockwise: frame system, cladding system, frame assembly, cladding assembly. Source: Author

This presented a conceptual problem in the exploration of group form. If its geometry is a gradient, the parts then become non-interchangeable. Thus, reconstruction or the metabolism of the group

[13] A knapsack is a resource allocation problem which, given a number of weighted elements and a container with allocated volume, asks how to most efficiently pack these elements into that container. See: (Demaine, 2011)
In the case of timber beams, the elements are target lengths and the container is the size of each timber piece.

form becomes unattainable. What compounded the problem was an early change in the nature of interaction- instead of using constituting parts as individual sensing and reacting units, the project was forced to be constrained to a much more simplified configuration.

Figure 16: Construction process from small hexagonal parts (left) to larger cladded assemblies (right)

The physical components that allowed the object to interact with the surrounding field are the cube core and tracking camera set up to overlook the surrounding context (*Figure 17*). The cube core contains a speaker and several LEDs in order to create sound and light-based output. The camera setup is an alternative to a sensor network configuration of the object's measuring capabilities where only a single camera was required to achieve awareness of the actors' presence in the proximity of the object. The advantages are that only a relatively simple setup was required. The disadvantage is that the sensory apparatus is no longer a part of the object itself, but an external system that had to be connected by external systems of connectors and wiring. The reduction of sensory and actuating capabilities to a single element meant that the possibilities of applications of these technologies and approaches were no longer tested in the context of the group form but instead, these systems could be conceptualised as operating from the point of view of a single part within the group form. Thus, design investigation 2 evolved into the prototyping of a conceptual component that is embedded in a virtual context of flows of information and functions to transition from physical presence of an object to virtual analysis and back to physical presence through interaction.

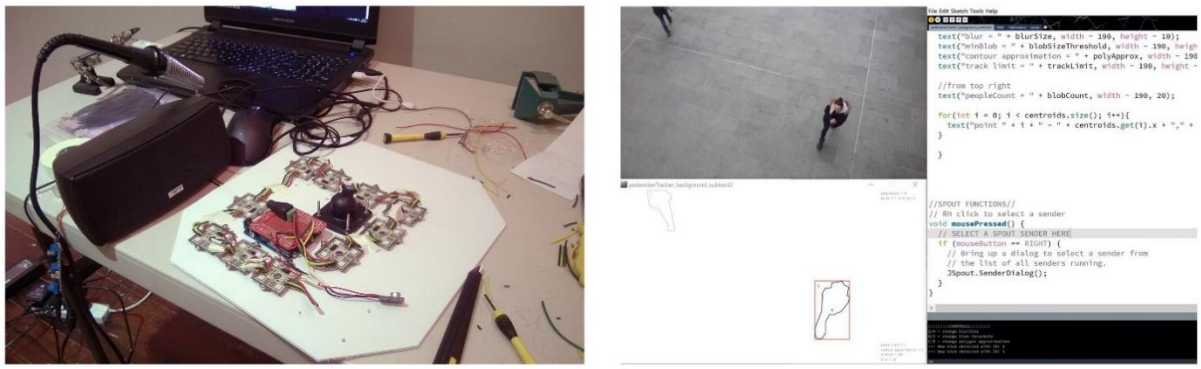


Figure 17: Assembly of the interactive cube core (left), Open CV- based Processing application (right)

5.3. Reactive System

Interactive system was developed out of the same mapping methodology and conceives of the surrounding perceivable context as a conceptual field. In order to create feedback for the form, locative media technologies were examined as the medium for the system. The idea behind locative media is to use geo-locating technologies that are context-aware and include GPS, RFID, sensor networks and computer vision to interface with physical bodies in space. [14] The process required a transition from analysis to an actuation mechanism- where field was a speculative simulation while the mechanism attempts to survey local space for precise (or close-to-precise) information and create appropriate responses. While presenting a number of possibilities for transforming a public space into a spatial technological interface similar to the one discussed by Weiser[15], this project attempted to tackle two distributed inputs: distributed network agents (social media posts) and local perceptible agents (people walking into the perceptible surrounding area or field).

Utilising computer vision libraries from openCV[16] and programming interaction with Processing[17], the system was set up to isolate the presence of actors within the field of vision of the group form *(Figure 18).* Robert Cameron's *Dead Reckoning Project* (DR) had previously experimented with the role of computer vision to create an alternate application for an urban screen (Cho & Macfarlaine, 2015). In this application, the software is set up to read the ground plane as a field of events where the actors (people moving through the field) are able to affect the behaviour of the form inside its broader

[14] Locative media refers to a *"range of experimental uses of geo-technologies including location-based games, artistic critique of surveillance technologies, experimental mapping, and spatial annotation."* (Hamilton, 2009, p. 393)

[15] *"Ubiquitous computers… reside in the human world and pose no barrier to personal interactions. If anything, the transparent connections that they offer between different locations and times may tend to bring communities together."* (Weiser, 1991, p. 104)

[16] openCV is an open source collection of computer vision libraries written in C++

[17] Processing is an open source programming language for creative programming

boundaries, creating the effect of mediation. Originally, we considered the use of projection mapping and an LED matrix wrapped around the object as ways of communicating to the users the state of the field. Due to time and budget constraints, these methods of feedback proved to be unattainable so instead, the object uses a 'responsive core', which provides feedback through sound and light. When the field is empty, the object begins to search for virtual presence of social media input. When it finds it, it activate the cube core which reads out text that it finds. Thus, the object switches between local field that is read by a camera and a virtual space of social media inputs. The reactive system of the field was achieved by combining two systems: system 1 and system 2 into a single reactive whole *(Figure 19)*.

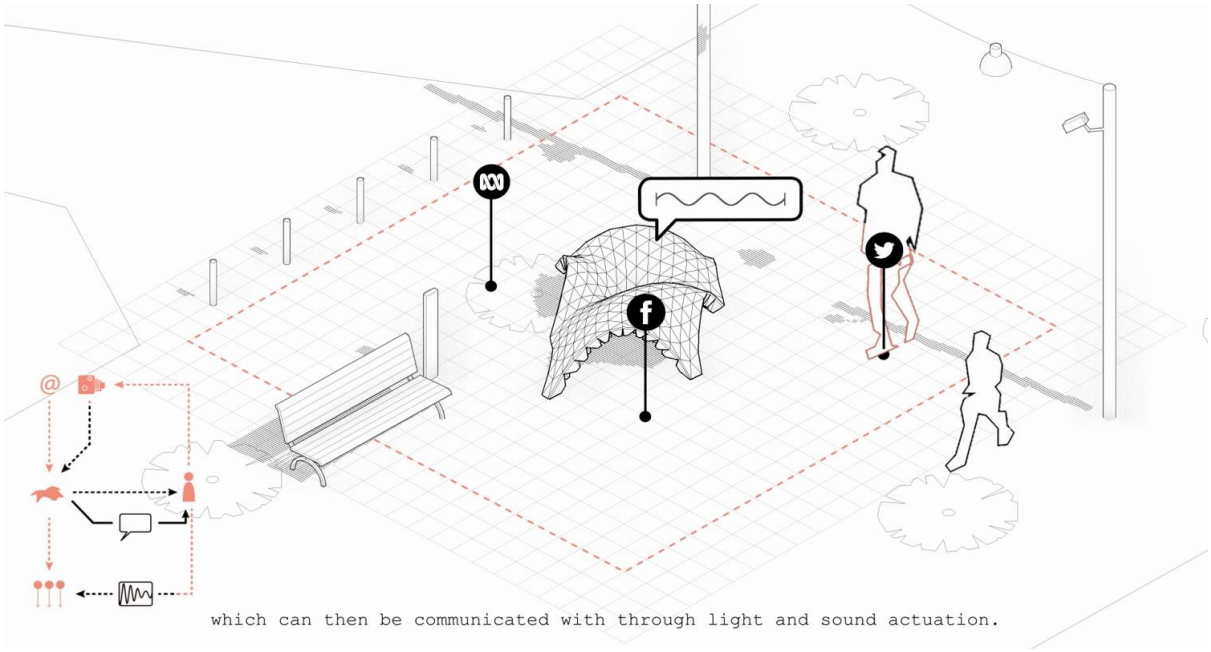


Figure 18: Stills from concept animation describing the field of vision of the object and its response behaviour. Source: Cameron R., Smolik A.

System 1 manages the social media scraping of the large-scale presence of people on site by restraining it to the event feedback. If a new online response is made, the system proceeds to create a text-to-speech response. System 2 manages local surrounding field of the object. The field of interaction is limited to a zone around the object, defined by the limits of the field of view of the tracking camera. The actors' position is calculated with computer vision software based on the camera input and depending on the distance from the object, the script decides whether to activate a response. The local field is given priority so that when system 2 detects presence, it suspends the social network inputs and awaits for the actors to reach close proximity to the object.

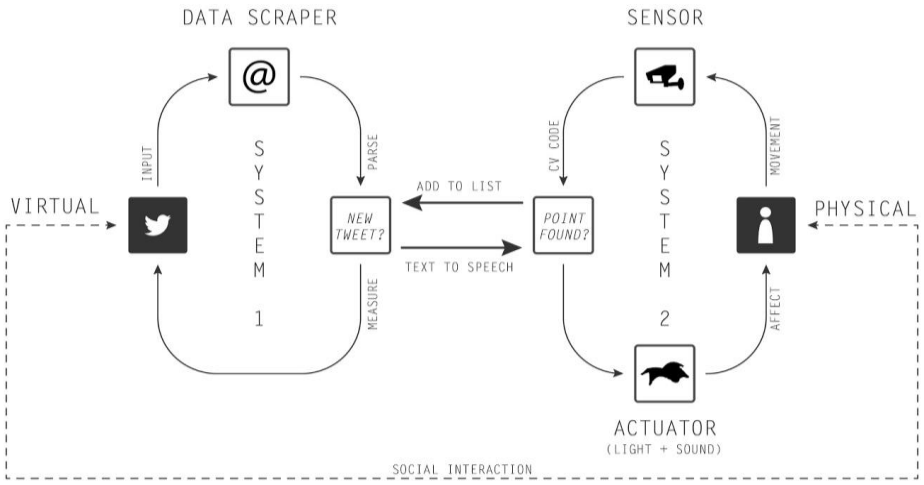
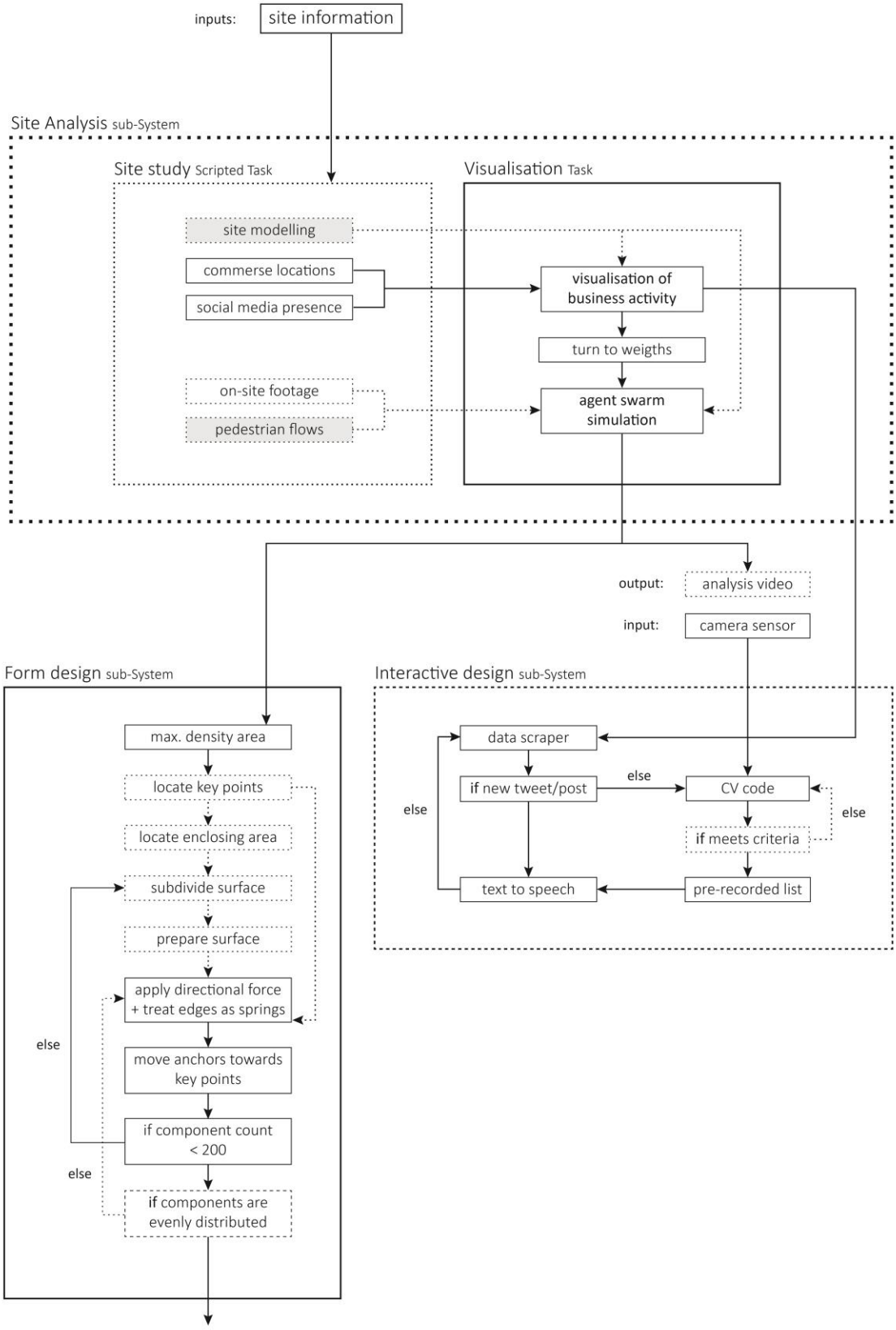


Figure 19: Systems design diagram. Source: Cameron R., Smolik A.

5.4. Summary & Discussion

Design investigation 2 examined group form's operational categories to *select* and to *mediate*. *Select* category involved an investigation into site analysis process that was looking at aspects of ubiquitous computers and speculative simulation. This method indicated that the context of the site as usually conceived through architecture has expended to incorporate a wider, non-physical set of interactions and experiences. This exploration demonstrated importance of further development of these tools to better understand techno-social aspects of any given context. *Mediate* category explored a non-physical medium for connectedness between parts in group form. Albeit, the process did not explicitly create group form, it explored an inverse top-down set of design parameters that need to be incorporated into the methodological process of its generation in order to impose external constraints on its 'growth'. This category also demonstrated that the secondary medium for group form elements could be conceived as a non-physical 'field' of interactivity that operates as a sensory input. This input can be further enhanced by the addition of multiple data sources that operate at different scales. In this case, computer vision provided necessary response from immediate surroundings and internet social media scraping which connected to the event but have not necessarily have been physically present. The effectiveness of the feedback that the form generated was another compromise. Instead of using tectonic elements of the whole as individual reactive and responsive elements, the design reduced responsiveness to a single unique part.



(continued on the next page)

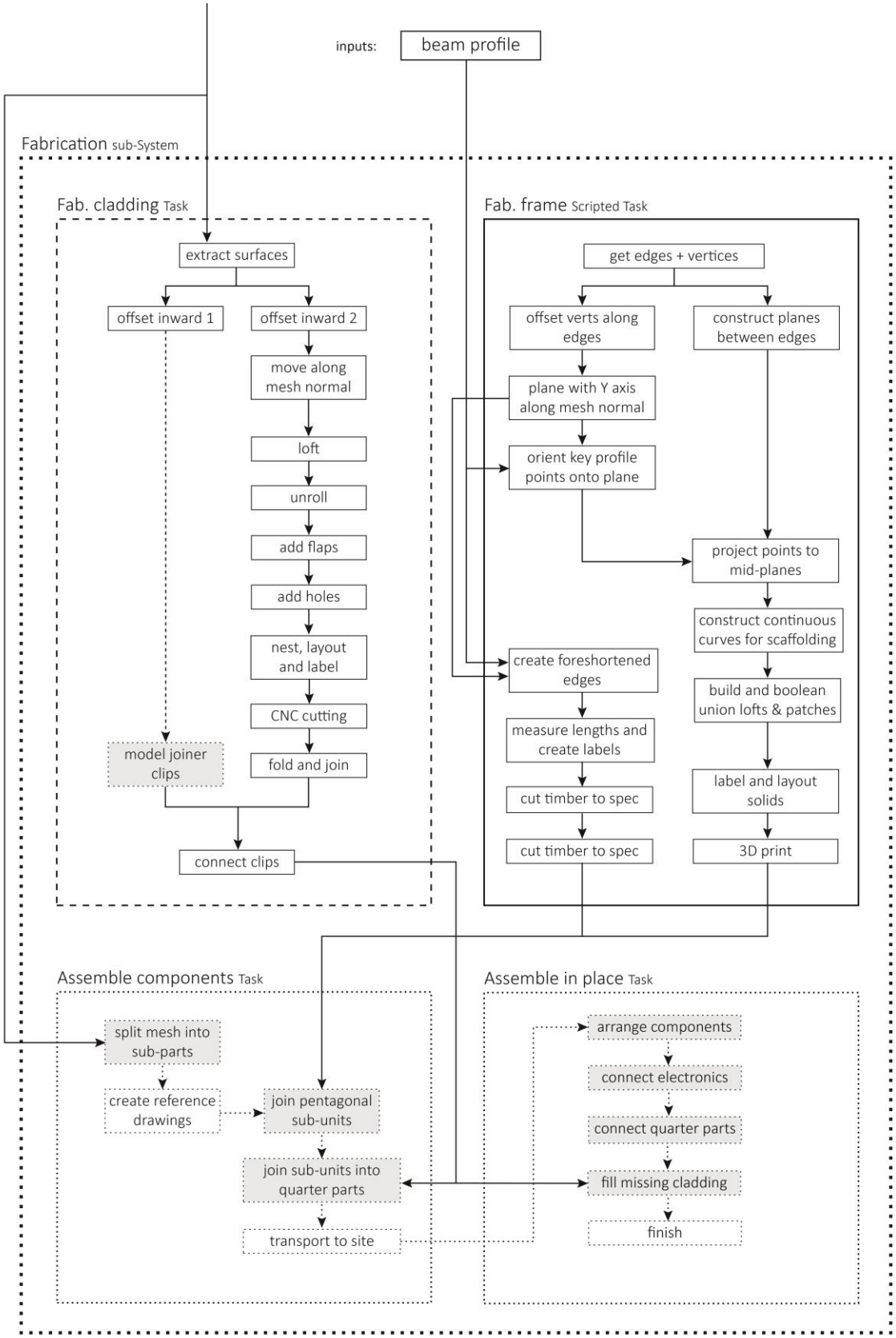
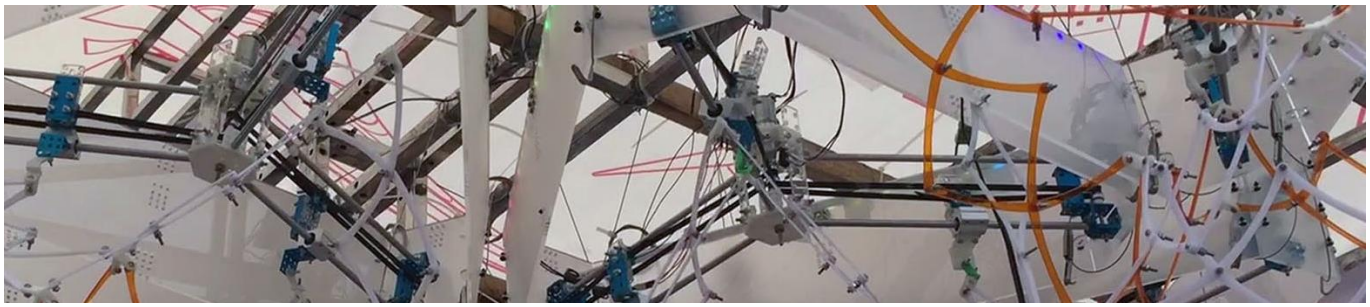


Figure 20: Modular Workflow Diagram for Design Investigation 2

Figure 21: Select + Mediate project in Claremont, Perth during Public 2016 exhibition

Even if we consider this investigation as being only an experimental 'part' of a potentially bigger group form, it still raises a question of the effects of interactivity of this part on itself and any additional aggregate structure. During the course of the event, interactive systems did not sufficiently contribute to the changes to the object *(Figure 20*)- they operated through the physical design to connect the actors to each other through the systems but not to the object. In order to create a closer relationship between the system and the object, the interactivity needs to incorporate another layer of public's ability to influence the morphology of the design, a system that enables this evolution to occur. This does not have to be an immediate mechanical change in its geometry- another possibility is to incorporate in the physical reactiveness of architecture the ability to suggest modifications that the user (or groups, collectively) can enact. At larger scales of buildings and urban design, these processes could be slower, taking effect over the course of extended time frames. In these scenarios, the top-down constraints would need to update, to take into account the evolution of the group form and the many constraints that are imposed on it.



Design Investigation3:

Kinetic Form

6. Design Investigation 3: Kinetic Form

In the introduction to Interactive Architecture, William Mitchell describes a hypothetical advanced architectural assembly able to self-replicate, respond to external stimuli, problem- solve and even evolve its form and programming*: "over time, this process generates large oyster beds-megastructures, composed of individual living pods, that might have been the envy of the Metabolists"* (Fox & Kemp, 2009, pp. 11, 12). This suggests a different paradigm for architecture that alters not only its function but its relation to the occupants and the role of the architect. In that sense, Maki's concept of architectural programming takes on a more literal sense where the system's regenerative and assembling properties that were originally conceptualised as a set of programs and procedures for construction and occupation to be implemented by users is enhanced by introduction of sensing and kinetic systems that allow architecture to adopt and 'grow' independently.

In light of this, the final investigation of this thesis focuses on exploration of direct effects of interaction upon the physical form of architecture and what this means for the design and conceptualisation of such systems in the context of current technologies and approaches. It also extends and builds on knowledge acquired from previous projects in order to better understand a broader range of potentials for digital fabrication and assembly techniques. This project began shortly after the completion of Investigation 1 and thus went through a longer process of design exploration, which began with the investigations of large parametric structural systems and evolved into looking at more interaction-based design approaches. This project also attempts to build on the knowledge of design investigation 2, which experimented with the idea of fields and influences through amplifying contextual feedback with the aid of social media and computer vision. Improvements to the design strategy from previous investigation that adopted morphology and interaction characteristics that did not change through interactions whereby the group form has no potential for adaptation or feedback involve direct experimentation with the possibilities of immediate, tactile interactivity in public space by deploying a relatively simple mechanism that operates in a relationship with the envelope.

To Repeat category was revisited in this project, incorporating responsive elements that were designed as a system that could change morphology of the group form. This investigation attempts to experiment with aggregation of individual elements similar to the *Cellular Form* project, with the difference being that these components' systems were designed as directly intractable through physically modifying components in real time, creating repetition with potentially dynamic consequences on the group form.

Creation of dynamic elements requires integration of various systems- geometric structural and elector-responsive as well as their interactive logic. The following chapters present a design and fabrication methodology which was explored to integrate kinetic behaviour with material constraints. The protocols developed in this project- how structure, interaction, materiality and geometry is impacted by assembling discrete "smart" components into larger, potentially indeterminate wholes and what challenges emerge and how we may be able to systematically control these in the future.

6.1. Responsive Design Strategy

The project was conceived as a modular, component-driven system, which exemplifies the concept of group form with responsive qualities. The end design goal is to create a self-supporting structure using reciprocal kinetic components that operate on computational protocols and allow it to exhibit self-organisational properties. The modules were experimented as parts of a component-driven systems construction based on a carrier-component surface geometry called responsive carrier-component envelope (RCCE). The project fundamentally seeks to understand the opportunities and consequences of how local components relate to the whole carrier envelope with multiple constraints and scale considerations on its structural, mechanical, responsive/interactive and tactile properties.

The forerunners of the concept of kinetic architecture are Willian Zuk and Robert H. Clark who in the seventies, envisioned kinetic parts and entire structures adapting to the needs of its users (Zuk & Clark, 1970). Around the same time in the 1970's, Nicholas Negroponte built a body of work around ideas of responsive architecture which integrated computation into spaces and structures through recognition, intention, contextual variation and meaningful computed responses (Sterk, 2003, p. 226). The major common feature between these approaches, as well as those speculated on by Maki, is the ability of architecture to adapt to a variety of states akin to an on-the-fly parametric model, modifying the morphology and enabling user's input with emergent situations having an impact on its states. This presents architecture with a more direct relationship between architecture, its occupants and their needs. In this sense, architecture here is defined as an integrated spatial strategy with interactive protocols, which can achieve a change in state given a series of inputs from its users.

Due to various advances in computing technology and fabrication as well as its greater availability for creative prototyping with platforms such as Arduino and Raspberry Pie, design and fabrication of such assemblies becomes feasibly for non-experts in fields of computation and electrical engineering.

The design strategy here is inverted- unlike previous projects, the interaction is developed from individual parts upwards into larger structures rather than subdividing these into smaller components

(Figure 22 & Figure 23). These components are made to interact at local scales with kinetic action animating its responses. The local interaction was envisioned to be able to communicate across the entire structure initially using wireless communication between parts and local algorithm capable of responding to local neighbours but due to various project constrains imposed by the complexity, scale and construction availability the communicative aspects of the project had to be scaled back.

Figure 22: initial design render and interactive scenario

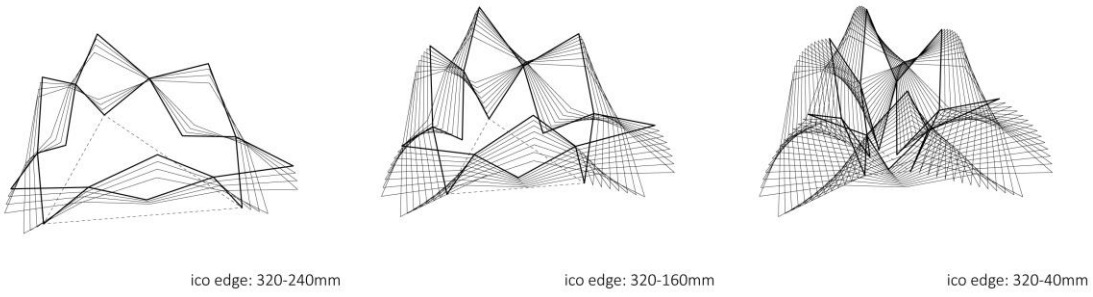


Figure 23: simulation of scissor truss retraction in Grasshopper

6.2. Digital Fabrication and Material Testing

The process of design of RCCE went through a series of phases where the synthesis between local reactive components and global structural possibilities was explored using a variety of parametric

approaches. The carrier component was initially conceptualised as a part, which has a responsive LED linked to a sensor.[18] This evolved into a kinetic structure as our understanding of electronic components improved. These components were to clad a carrier envelope- a continuous structural shell housing all the necessary functional parts of the design. Various permutations were developed- some from previous projects, other after the Kinetic Form was constructed and based on methods developed elsewhere. These structural systems essentially break into three main categories: skeletal planar piece fabrication, spider joint fabrication and finger joint cladding fabrication.

Planar free form fabrication (PFF) follows a method described by Sevtsuk and Kavlo (Sevtsuk & Kavlo, 2014) which takes a mesh of arbitrary topology and proceeds to generate beams (oriented to average edge normals) connected to vertex joints that 'sandwich' incoming channels between two planar star-shaped parts. This method is particularly useful when trying to create complex forms from planar parts however, the total number of unique parts is particularly large which complicates construction.

Adaptive spider joint method (ASJ) is similar to methods of frame construction described in previous projects. Here it was significantly modified allowing for an arbitrary channel profile input, custom tolerance, chamfer radius, rectangular outside profile (in case of chamfer radius=0) as well as a merge function. The merge function allows joints in close proximity to be joined together, minimising the total number of beams required. This method improves the number of unique parts required for construction by maintaining a standard beam profile and culling small spans where necessary.

Finally, the adaptive finger joint fabrication method (AFJ) is a slightly different approach in that it is not a method for generating frame-like structures but instead it creates planar tessellation by inserting finger joints between neighbouring panels. It has a number of notable constrains in that the input discrete parts must first be flat (which can be achieved by iteratively modifying shapes with springs using Kangaroo plugin) and second, the number of edges meeting at a vertex must not exceed three. This is due to the arbitrary directionality of material thickening at vertices. This method, if developed further, has the potential for acting as a cladding technique where interactive assembly is imbedded in flat faces of the structure without the need for continuous variation in geometry of interactive assembly parts.

Figure show both physical and virtual models constructed in order to test these various strategies for constructing a RCCE. Several scaled physical models were built of PFF and AFJ to test for their durability and fabrication complexity. The former proved to be extremely laborious even for simpler

[18] See Appendix part D: Visual diary – progress images for a visual description of the design development.

geometries (such as the hyperboloid-like structure shown). The ASJ and AFJ methods were more promising but their coding took too long to implement and wasn't used in the final assembly.

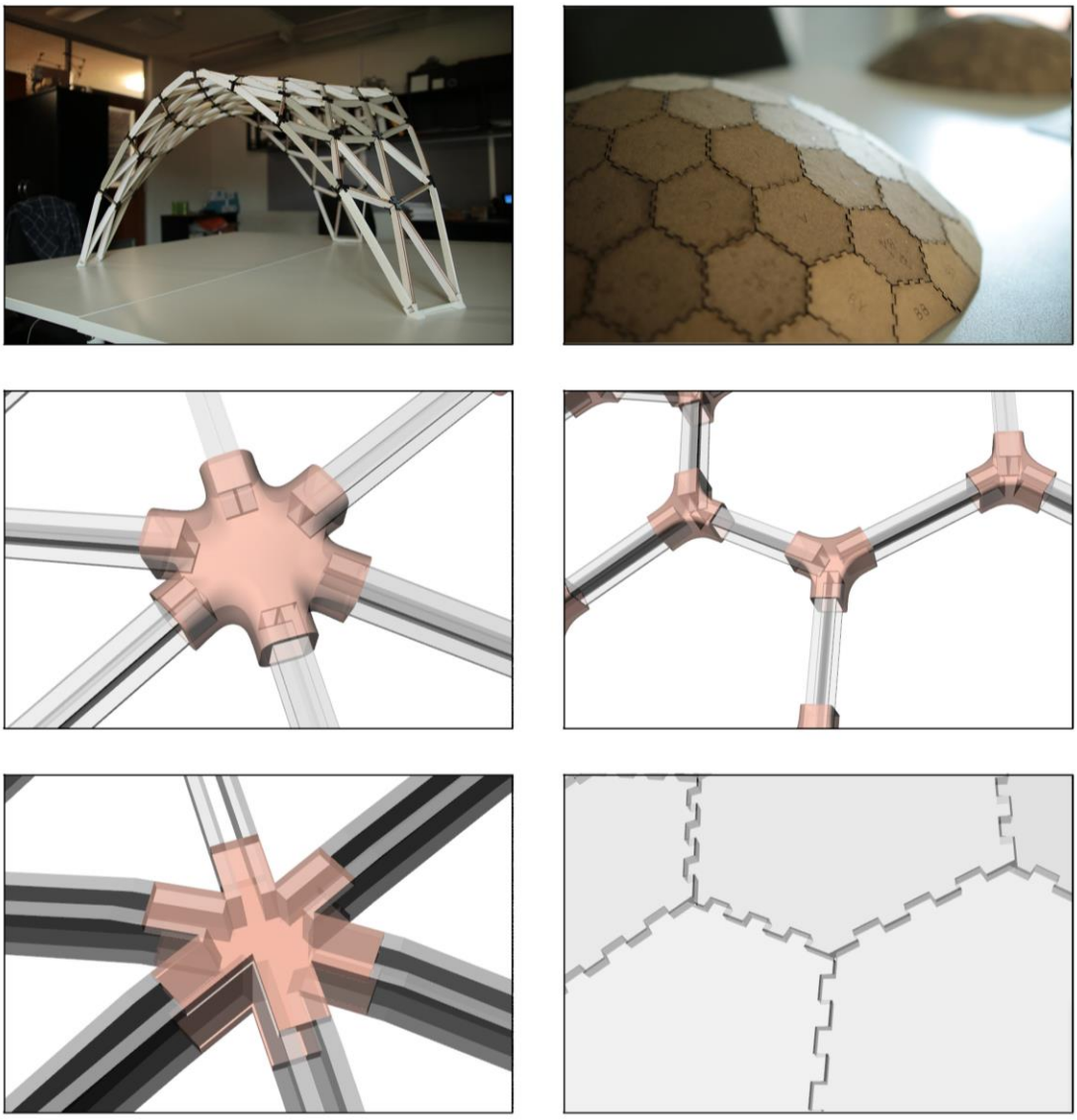


Figure 24: Various models of the three key methods of free-form construction developed for Kinetic Form (clockwise from top left): freeform surface fabrication as described by Sevtsuk and Kavlo, adaptive finger joint fabrication physical test model, parametric spider joints developed from earlier attempts at framed structure fabrication with 6 nodes per vertex and 3 nodes per vertex, PFFS digital parametric model and finally, a digital parametric model of AFJ.

The creation of a responsive component design developed in parallel with RCCE. It was assumed that once local geometry was found, it would be possible to integrate it with RCCE. What was found, however, is that this synthesis of part and whole is more problematic than originally expected. The main complication is the continuously variable overall structure that necessitates the use of unique components. Due to complexity of construction and the overall number of parts required for each

kinetic component, the upward pressure of this process necessitates self-similarity of parts and thus, a homogenous RCCE is required without any changes in convexity or singularities in the tessellating pattern. The way a generic component was conceptualised is in three main parts- Structural Frame is the skeletal mount of the part that holds together the mechanism and allows it to connect to other parts. Guiding Assembly is the kinetic mechanism and its mechanical components that allows the part to sense and respond to its environment. Kinetic Truss is the moving geometry of the assembly and not directly connected to the structure but exists as a secondary cladding. *Figure* 25 shows all parts of this assembly as well as some design iterations of the Guiding Assembly which was improved to use standardised components which were previously 3D printed in the prototype stage.

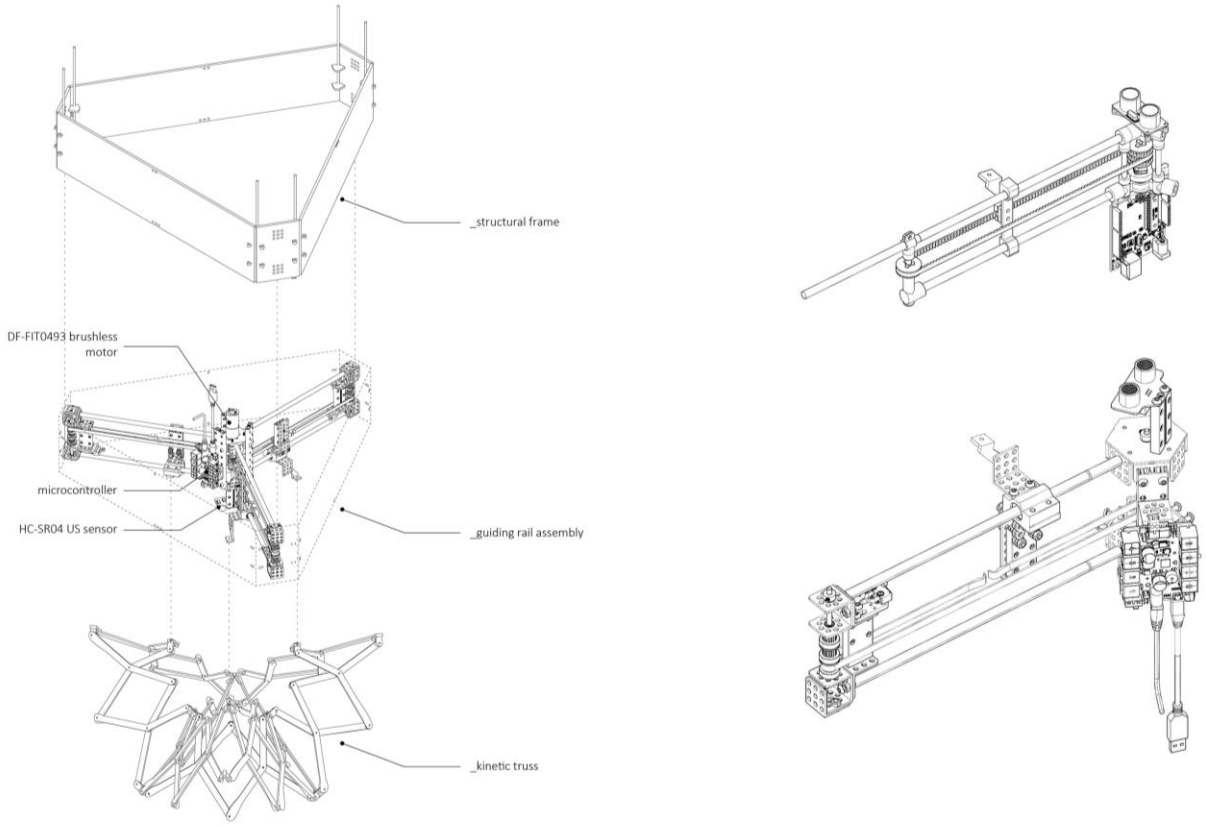


Figure 25: multi-layered assembly diagram, right- development of the motor actuation system

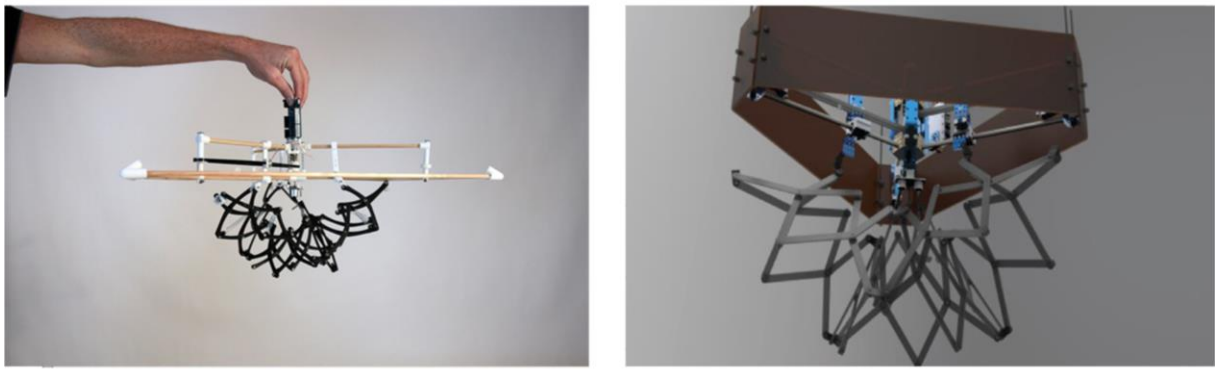


Figure 26: component physical prototypes: left- original design test, right- component using standardised components

6.3. Reactive System

The intelligence of the installation is based on the concept of *Subsumption Architecture* described by Rodney A. Brooks in the paper titled "Intelligence without Representation". Interaction and creation of intelligent systems is conceived through construction of robotic agents called *Creatures* that when aggregated into larger systems display emergent behaviour due to their interaction with the environment which is also hypothesised to operate at human level of cognitive activity without detailed representations[19].

The component reactive system is composed of sensory input, processing algorithm stored on a microcontroller and kinetic actuation system. The sensory input stream, enabled by a HC-SR04 ultrasonic sensor is able to detect simple changes in its environment. This simple input does not require a complex representation and the ability to reason about it- the reaction from input is easily computed, allowing for a sensing of the environment that can occur frequently, giving the component the most up-to-date model of its context.

The group form is distributed- it has multiple parallel activities with no central representation. This reduces the chances of a total collapse of the system due to sensor noise, incorrect processing or mechanical failure. Each one of these 'layers' could be thought of as having a re-configurable logic- the hardware could be matched to specific goals existing in parallel. There is no central control, only a distributed network of components.

[19] Representation here refers to abstract models used to construct robotic or artificially intelligent systems. These models, as Brook argues are insufficient for constructing sophisticated artificial behaviours able to deal with unexpected changes in its external environment for a variety of reasons. For complete explanation see: Redney Brooks.

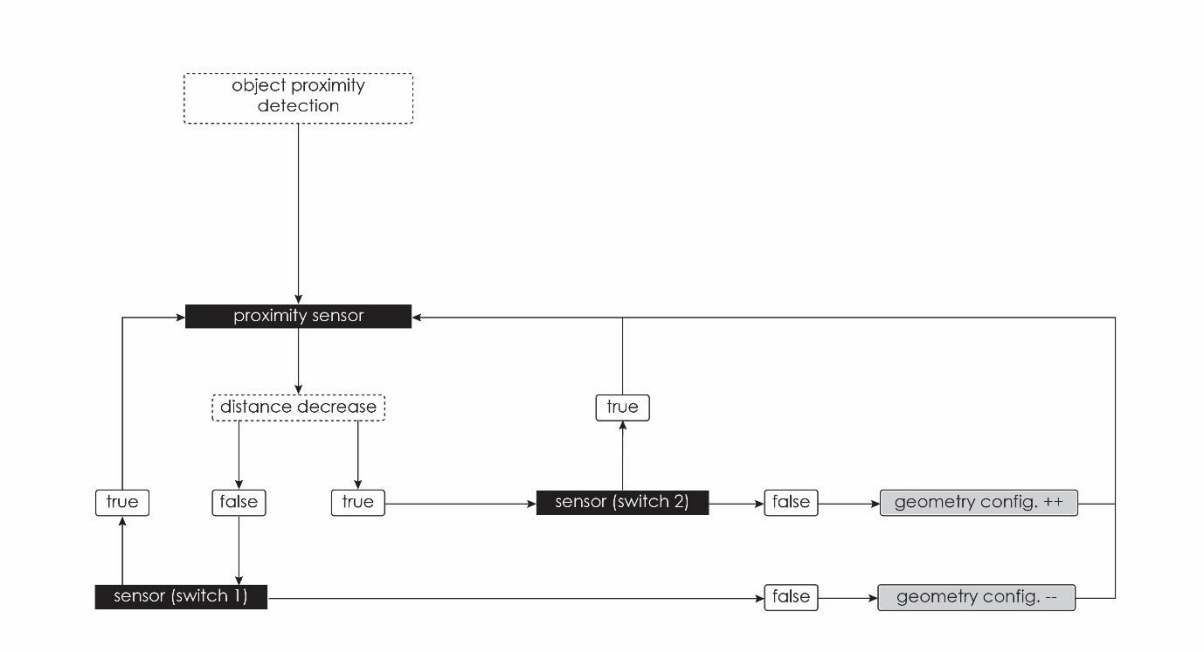


Figure 27: pseudo-code of interactivity

The system described here would be a “reactive” one and not “interactive” due to a non-circular interaction. A truly interactive system will be containing many loops and is capable of maintaining a conversation: a continual and constructive information exchange. Marcos Novak uses the term “transactive intelligence” to describe architectural system that not only interacts, but transforms the user and itself (Silva, 2006). This level of intelligence would be necessary to create a prototype for a feedback system required by the group form. Due to this, a custom interactivity interface plugin for Rhino and Arduino began development (*Figure 28*) to address the issues of interactivity that is too simple to create complex feedback loops. Due to time constraints, this plugin was put on hold.

Figure 28: plugin in development as a result of investigation 3

6.4. Summary & Discussion

Figure 29: Shenzhen event- public interaction with the project

The project experimented with interactivity as it applies to group form and the idea of component-based architectural assemblies. The principal motivation was to explore design workflows, modelling and digital fabrication technologies as they apply to kinetic actuation and interaction. Collaborative aspects that required working with an international team as well as installation and display during a public event ensured that the process approximated real-world conditions where collaborative aspects and scheduling are of importance.

The project implemented reactive interactivity, whereby people's responses were directly translated into kinetic motion by the structure. The project suffered from a lack of resources and time which limited its scope as a fully flashed out exploration into the interactive aspects of group form. Future work based should explore a wider range of possibilities. Approaches such as those described by Rosenberg in *Designing for Uncertainty: A Novel Shapes and Behaviours using Scissor-Pair Transformable Structures*(Rosenberg, 2009) give a broader view on a potential variety of physical systems and interactivity approaches which should be explored in this area.

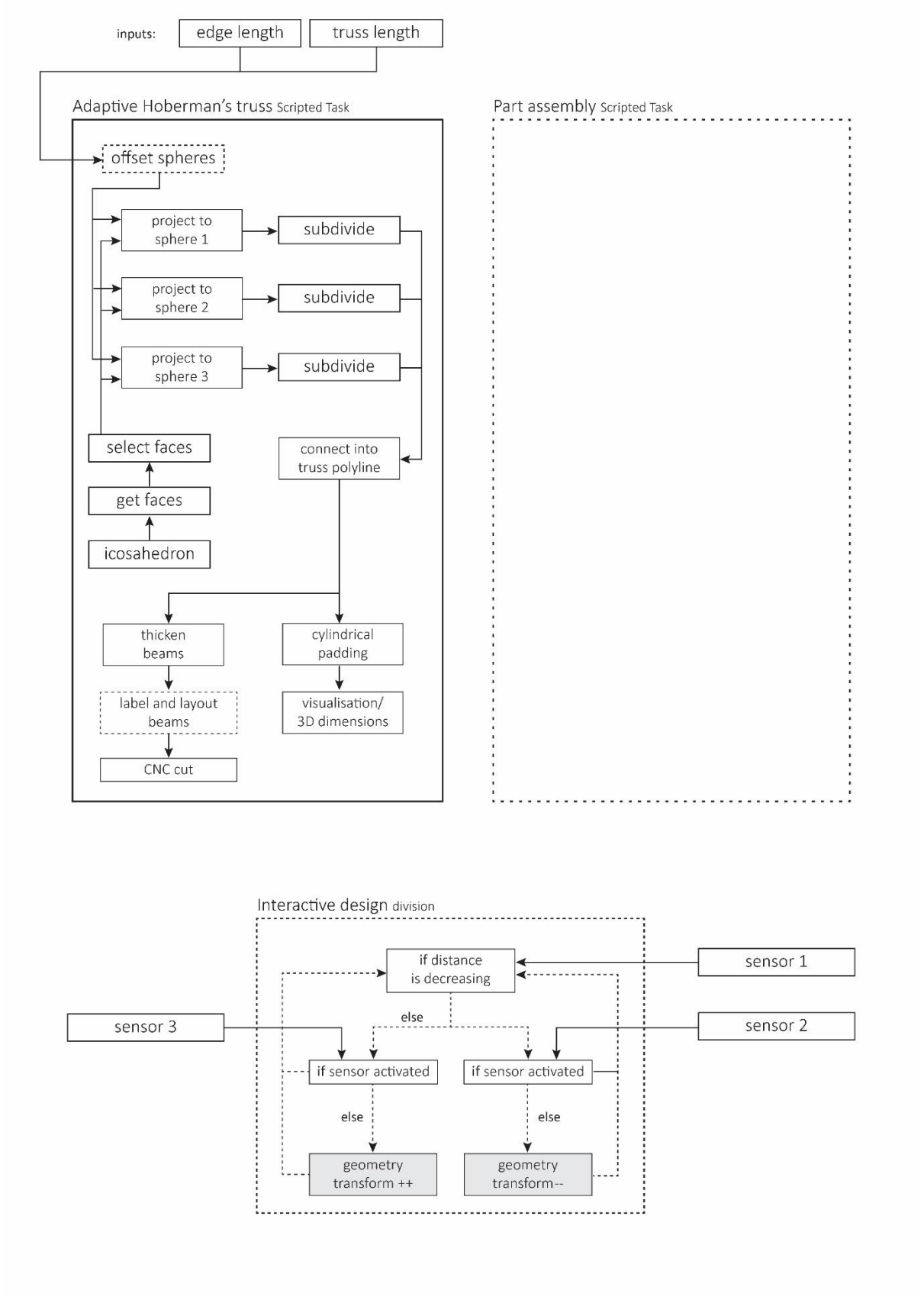


Figure 31: Modular Workflow Diagram for Design Investigation 3

7. Form to Behaviour

This research set out to interrogate emerging practices within architecture, informed by the theory of group form as well as contemporary techniques of conventional and non-conventional material and digital experimentation with simulation, digital fabrication, assembly and prototyping. The focus is on interrogating an approach to design enabled by modern tools of conceptualisation, communication and realisation and informed by Maki's theory of group form which, originally aspiring to develop a new method of designing assemblies approaching the scale of building clusters and even cities, found a broader conceptual application to a wider range of scales. The focus of practice fits the aspirational nature of group form, which aspires to a more experimental approach driven by exploration of relationships between parts, design of systems and modular synthesis of both form and process.

This interrogation of design process, done in the context of an existing design theory of group form that is often overlooked in praxis, does not aim to generalise design methodology. This would inevitably put unrealistic constraints on designers and will be resisted (Willis and Woodward, 2010, 201) due to design's inherent emergent and spontaneous quality. Group form itself was conceptualised as a reaction against these generalisations contained in tenants of modernism and related urban theories, which developed out of it. Rather than prescribing a specific methodology, this thesis- like the group form formulation, attempts to experiment with modularity, system design and context with the aim of extending architectural vocabulary in terms of form and process.

7.1. Group Form as Process Modularity

Through the project chapters, varying degrees of process detail were discussed. In chapter *3.2 Workflow Hierarchy* I described some methods for analysing design processes developed by John Everett and Nicholas Williams on which the four-part scale model of this thesis is based on. It is designed to address the primary aim of this research, demonstrating the idea of Group Form as a modular process rather than modular form. The projects in this thesis explore finer levels of detail at the scale of fabrication and assembly, construction of simple automated systems and design decisions and workflows that combine digital models of fabrication and modelling with manual tasks and approaches. This is done in order to illustrate specific points at which these 'modules' couple. This gives an insight into the 'level of modularity' of these systems: if they are able to recombine into new systems and what are the thresholds for this reconstruction. The focus of process pushes design

prototyping beyond unique, idiosyncratic objects towards an epistemological approach that is able to cope with construction of new, increasingly complex assemblies with modular and increasingly automated systems. Through the process of exploring this taxonomy as applicable to design and fabrication of prototypes presented in this thesis, it was discovered that that these models are highly subjective in the way they classify level of process detail categories. The finer levels may vary in complexity, while categories themselves may 'drift' across scales depending on the description of a given process in a specific design context. This section attempts to address these inconsistencies, suggest some approaches that might improve classification rigour and quality of analysis of design processes.

Beyond simply describing the modular process, I wish to address the nature of this formulation of group form. While analysing project material presented in this thesis, certain patterns emerge in relation to the idea of process patterns, their modularity, coupling and specific design information flows. One obvious observation is the inverse relationship between the level of detail and the degree of modularity- as systems increase in complexity, it becomes harder to integrate them into new contexts. The level of transparency of these systems also scales up so that lower-level details are more intelligible and thus re-usable while higher-level ones become increasingly abstract and require higher levels of expertise and time in order to implement and adapt to different contexts.

7.2. Division

Unlike the organisational patterns of other categories that emerged in prototypes presented in this thesis, the number and general functional types of Division categories remain relatively stable across all projects. This is predominantly due to the top-down approach where the structure of these categories for each prototype was decided before the commencement of work. Due to relatively few external constraints, a very significant control over the high-level process was achieved. These categories were guiding the rest of the process around which lower-level assemblies of processes formed often in formations that were far less predictable. The main Divisions that were addressed are Form Finding, Fabrication, Site Analysis and Interaction. In every project the Interaction Division consisted not of Activities but Tasks and Basic Methods, skipping the entire category altogether. This is likely due to a relative simplicity and small physical scale of interactive systems that were able to be resolved in simple Divisions that demonstrates a level of fluidity in the hierarchy framework where

the path toward high complexity categories ought not to necessarily contain all of the sub-categories present in the framework.

In terms of modularity of process, Divisions are highly non-modular in design process. The only highly modular Division presented here is a From Finding Division from Mediated Form. This is predominantly due to a relatively widespread use of spring systems in the form finding process with tools like Kangaroo for Grasshopper allowing the Division process in this case to become relatively modular. It still allows a wide degree of adjustment internal to this category, however design process at this scale required not so much a high level of adjustability as the modularity of its internal processes altogether in order to achieve bespoke formal design outcomes. This internal structure of Divisions would necessity increase exponentially with the level of external and often intractable constraints imposed by more complex structures with multiple additional structural, mechanical and functional constraints. By introducing unique types of Divisions, the coupling between these categories also seems to increase as in the case of Mediated Form where the Interaction Division is strongly coupled with Site Analysis decreasing modularity of both.

7.3. Activity

In commercial construction, Activities is the basis for modular production of construction components based on standard products and systems with fabrication typically accomplished by the supplier. In the context of bespoke production at this level of process detail involving external manufacturers and fabricators, well controlled and minimal sets of information is required for a high degree of modularity (Williams, 2017, p. 187). Examples of such systems of bespoke process control and communication with suppliers are described in Aggregated Form and Mediated Form projects where CNC and 3D printed parts needed to be manufactured by external fabricators offsite. This process is assisted by automation of fabrication tools used by these manufacturers often deploying familiar file formats and conventions. Kinetic Form project communicated with a remote team to communicate assembly instructions of parts and the process of fabrication available to them that revolved largely around digital 3D models that described arrangements of parts, whether they consist of standardised components or are fabricated on site using methods of rapid prototyping. Most of the Activities in this research largely consist of in-house to allow direct access to design and testing of bespoke solutions at the level of process detail of activities.

In terms of the taxonomy of the level of process detail, it often sits somewhere just above the level of automation. Often, the Activity task would involve operation of a piece of software or a particular

script in order to achieve a certain outcome. This outcome could be either physical (some component that needs to be manufactured) or digital (simulation or form finding). In cases of interactive design, Activities disappeared entirely as a descriptive category although in some cases they involved automation of other relatively complex systems such as the computer vision code in Mediated Form or the sensor and actuation assembly of the Kinetic Form projects. In cases of interaction, the process Division or Activity became the final outcome rather than serving as a support tool for analysis or fabrication in other types of Activities/Divisions. This might also be due to our relative naiveté regarding interaction systems in built structures and their possibilities that consequently forces us to construct relatively simplified systems of control for these Divisions, bypassing Activity categories altogether.

7.4. Task

Tasks are understood in the level of process detail scheme as a kind of automation threshold category with no definitive boundary. Everett suggests that they go beyond the specific examples of processes that can be performed by single robots as *"mutually exclusive and collectively exhaustive for all on-site work"* (Everett, 1991, p. 69). Williams considers Tasks as almost a completely 'automated category' in principle by concluding that some Tasks cannot be easily automated and that difficult and complex decisions must, sometimes be embedded in algorithms but that this, could potentially be done using more advanced approaches such as machine learning (Williams, 2017, p. 187). This however opens an upper limit on this category and raises some questions regarding this (and other) categories in the level of process detail taxonomy. Instead of asking: *"can higher levels of this taxonomy be automated with some more advanced computational tool/s?",* a more pertinent question that arose in the course of this research is *"what are complexity boundaries that separate these categories in the first place?*". For higher level categories this separation arose primarily from design context where processes can be parsed relatively easily by looking at neighbouring categories and their role in the design process. These lower-level categories, however, are too general to be understood in this way.

As stated in section *3.2 Workflow Hierarchy*, the differences between Task and Basic categories are not made clear enough in the literature with the work in this research further highlighting problems with more granular categories in this taxonomy. Throughout diagrams of projects presented here, Tasks and Basic Methods are shown as singular process components (shown as boxes in diagrams). However, complexity of these components varies radically from functions that use simple methods to calculate, for example distances between objects to full- blown 'scripts' or 'methods' (as well as

collection of methods, classes and even entire programs) that perform very complex operations involving multiple inputs such as camera feeds (as in the case of Mediated Form interaction Tasks). Due to this range, it is difficult to pinpoint the exact threshold that separates Tasks from Basic Methods (or Elemental Motions or Functions).

However, if we simply consider 'Tasks' as the simplest action performed in the process of design, we could stipulate that these processes describe certain low-level processes that in many cases considered for automation through scripting. With a wider selection of scripting languages and digital work environments, automating small design routines and fabrication procedures is relatively commonplace amongst large architectural practice and research. It has already been shown that simple organisation of processes at this scale improves intelligibility (here referring to legibility of scripts) that have implications on its modularity (modification of processes at this scale and ability to share and modify them across multiple users) (Davis 2012).

7.5. Basic Method

Basic Method or Elemental Function or Motion in literature is defined by its simplicity. This research demonstrates issues with this categorisation. Williams described this class as being made apparent by digital design environments such as Grasshopper. It is assumed that what is being referred to here are batteries- modular bits of code that have explicit inputs and outputs that a designer can rearrange in a way that suits desired requirements. The issue, closely related to the taxonomy of the Task category is that these batteries contain unspecified complexity. In fact, this complexity ranges widely from battery to battery and with the ability to download third party plugins and create custom code, this distinction is essentially without a difference. Moreover, it does not disappear at the level of code either. Since most of it is made from methods (the building blocks of object-oriented programming), different combinations of methods in batteries (classes) as well as the internal structure of methods themselves varies significantly adding to this the ability to reference software's own methods and local (as well as external) libraries creates a very complex network of process that cannot be easily classified in terms of its category. Williams also refers to disappearance of functionality at the level of Elemental Functions or Basic Methods. This, however, is also project and design-dependent and highly depends on which methods are considered functionally sufficient for a given task and which ones need to be written using different approaches and design intentions.

8. Conclusions

This information-based approach thus has consequences on the physical product of design which now acquires the capacity to “remember” discrete elements of process that illuminate complex relationships between form, material, fabrication and performance. It is also conceivable to imagine a temporal effect that this approach might have on the “metabolism” of the design object: if it has memory and a set of behaviours or instructions, which it executes itself or delegates to its users, it has the ability to adapt and develop in a way conceived of by Maki. It also provides a more immediate calibration tool for the internal logic of the design process and its interacting parts. If we can develop increasingly formal ways to describe design processes with all of its gaps and designers’ necessary intuition and uncertainty, we can begin to develop and perform meta-analysis on this system to glean certain tendencies, affordances, opportunities for automation and simplification, better workflows and new design knowledge.

8.1. Assumptions

The research project assumed that the conceptual frameworks developed by Maki and others could be extended through the applications of new methods and technologies that have emerged in the last few decades particularly digital fabrication and interactive prototyping platforms.

A multi-scalar analytical framework was introduced to ground the research which focused on systematic descriptions of design workflows and detailed overview of processes and methods used on the individual prototypes. This was hoped to ground research in a framework more conducive to analysis and rigorous interrogation. It was also assumed to have impact on a more systematic approach to studies of assemblies described in *group form,* particularly while utilising a design-centred or reflection-in-action methodologies involving small-scale prototype installations.

8.2. Contributions

The thesis explored several avenues relating to areas in design applications of digital fabrication and interaction of architectural form. Multiple design approaches were explored and their relationship to the group form were attempted to be explicated (see Table 2).

Table 2: Key hypothesis and questions formulated for each project:

Design Investigation	Hypothesis on form and performance	Key Question	Design Approaches
Cellular Form	Cellular collective form can be translated from virtual aggregating simulation into a material prototype	What are the challenges in creating aggregative group form (bottom-up) with a grafted system?	• Recursive scripting • 3D printing (Fused deposition modelling) • Simple interactive prototyping
Mediated Form	Collective form can connect and mediate context beyond its immediate physical presence	How can physical forces and locative technologies influence design parameters for group form?	• Site analysis and visualisation through scripting and simulation • Construction of complex interactive system • 3D printing (Fused deposition modelling), • 3 Axis CNC cutter
Kinetic Form	Collective form can be composed of kinetically-responsive elements	What are the effects on scalability and top-down control of form in group form with kinetic-responsive parts?	• 3D printing (Fused deposition modelling • Interactive Arduino Prototyping with kinetic elements

8.3. Constraints on the Results

One of the key issues encountered in this thesis is that of temporality- projects did not evolve over time and this was not envisioned in planning for most of the investigations. Installations presented in this research were only partially re-configurable (for example, cellular form was re-assembled after the exhibition into a different, smaller form). Even though this re-assembly was enabled by the modular approach, it was not central to the installation's focus and the other two projects did not have this as a feature due to their limited modularity and assembly procedure. Thus, the investigations did not evolve over time and in this sense, significantly limited the Group Form approach.

Another constraint was the scale of investigations diverged quite significantly from the scale described as group form that typically involved clusters of buildings and city-scale investigations. This

was done to prioritise prototyped investigations in digital fabrication and interactivity at a much reduced scale and scope.

Project 3 suffered from the lack of development and thus appears to be significantly underdeveloped in the context of the other two projects. This was due to constraints on the resources, time and difficulties of communicating with a remote team of people responsible for assembly and installation.

8.4. Future Work

There are a number of possible trajectories for future research based on the work in this thesis. One is a broader goal of developing a rigorous system for creating workflow descriptions or "master program". As opposed to using these in order to analyse exploratory design projects with no set brief or criteria, a more focused approach would be to attempt to create a general theory for generating them based on a limited set of specifically chosen criteria that relates to controlled aspects of projects. These aspects could incorporate performance, typology, urban characteristics and specific technological applications. The transformative or "metabolic" properties of such systems could be explored more effectively on larger scales, substituting processes that are based on speculative technological advances to ones that rely more on quantitative models of social, economic or cultural forces driving potential changes in simulated form, guided by master program.

Another possible trajectory for future research is to focus on specific material system application in the context of self-assembly. This would involve research into currently developing material or fabrication technology which has potential for group form applications. As technology improves, and it becomes possible to construct smaller and smaller synthetic assemblies, future parts of group form can be built from nanotechnological and bio-nanotechnological means. As Michael Fox and Miles Kemp note in their book *Interactive Architecture*: *"designers in the future will simultaneously develop the movement, method of connection, geometry, and the embedded intelligence of these smart objects."* (Fox & Kemp, 2009, p. 229). This embedded intelligence can be understood as the "master program" of group form. Printing out mechanical and electric parts in one seamless whole as parts become smaller and more intricate, the problem of design research becomes integration of various conflicting engineering and design concerns through a shared process description.

Bibliography

Abelson, H., 1986. *Structure and Interpretation of Computer Programs.* [Online]
Available at: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/video-lectures/1a-overview-and-introduction-to-lisp/
[Accessed 01 12 2017].

Abelson, H., Sussman, G. J. & Sussman, J., 1996. *Structure and Interpretation of Computer Programs.* 2nd ed. Cambridge: MIT Press.

Allen, S., 1999. *Points and Lines.* New York: Princeton Architectural Press.

Allen, S., 2011. *Discussions in Architecture: Stan Allen with Preston Scott Cohen* [Interview] (22 October 2011).

Aristitle, 2015. *On The Parts of Animals.* [Online]
Available at: https://ebooks.adelaide.edu.au/a/aristotle/parts/index.html
[Accessed 12 01 2017].

Aristotle, 1961. *Parts of Animals, Movement of Animals, Progression of Animals.* London: Willian Heinemann Ltd..

Becker, J., 2012. *SFMOMA Open Space.* [Online]
Available at: https://openspace.sfmoma.org/2012/10/field-conditions-i/
[Accessed 23 4 2018].

Bing, F., 1971. The History of the Word 'Metabolism'. *Journal of the History of Medicine and Allied Sciences,* 26(2), pp. 158-80.

Burry, M., 2011. *Scripting Cultures.* West Sussex: John Wiley & Sons Ltd.

Cho, M. & Macfarlaine, F., 2015. Perth Cultural Centre-Urban Life Laboratory. *The Architect,* pp. 25-28.

Collins, G. R., 1959. Linear Planning Throughout the World. *Journal of the Society of Architectural Historians,* 18(3), pp. 74-93.

Conference, B. A. C., 1965. *Architecture and the Computer.* Boston, Boston Architectural Centre.

Cook, P., 1990. *Archigram.* New York: Princeton Architectural Press.

Cross, N., 2006. *Designerly Ways of knowing.* London: Springer-Verlag London Limited.

Dade-Robertson, M., 2013. Architectural User Interfaces: THemes, Trends and Directions in the Evolution of Architectural Design and Human-Computer Interaction. *International Journal of Architecutral Computing,* 11(1), pp. 1-9.

Demaine, E., 2011. *Mit Open Courseware.* [Online]
Available at: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/lecture-21-dp-iii-parenthesization-edit-distance-knapsack/
[Accessed 12 4 2017].

Evans, O., 1999. Alternating Steady State in One-Dimensional Flocking. *Journal of Physics A: Mathematical and General,* 32(8), pp. L99-L105.

Everett, J., 1991. *Construction Automation: Basic Task Selection and the Development of Cranium.* Cambridge, Massachusetts: MIT.

Foster, J. B., 1999. Marx's Theory of Metabolic Rift: Classical Foundations for the Environmental Sociology. *American Journal of Sociology,* 105(2), pp. 366-405.

Fox, M. & Kemp, M., 2009. *Interactive Architecture.* New York: Princeton Architectural Press.

Frazer, J., 1995. *An Evolutionary Architecture.* London: Architectural Association.

Gandy, M., 2004. Rethinking Urban Metabolism: Water, Space, and the Modern City. *City,* 8(3), pp. 363-379.

Ganti, T., 2003. *The Principles of Life.* Oxford: Oxford University Press.

Hamilton, J. C., 2009. *Ourplace: THe Convergence of Locative Media and Online Participatory Culture.* Melbourne, Victoria, In: The Proceedings of OZCHI.

Harper, D., 2001-2017. *Online Etimology Dictionary.* [Online]
Available at: http://www.etymonline.com/index.php?term=metabolism
[Accessed 7 January 2017].

Hayward, T., 1994. *Ecological Thought.* Cambridge, Mass.: Polity.

Hensel, M., 2006. (Synthetic) Life Architectures: Remifications and Potentials of a Literal Biological Paradigm for Architectural Design. *Techniques and Technologies in Morphogenetic Design,* May-June, 76(2), pp. 6-9.

Hoberman, C., 1990. *Radial Expansion.Retraction Truss Structures.* US, Patent No. 4942700.

Hooke, R., 1676. *A Description of Helioscopes and Some Other Instruments Made by Robert Hooke, Fellow of The Royal Society.* London: T.R. for John Martyn.

Lawson, B., 1990. *How Designers Think: Design Process Demystified.* 2 ed. London: Butterworth Architecture.

Leroi, A. M. F., 2010. *Function and Constraint in Aristotle and Evolutionary Theory. Was ist 'Leben'? Aristoteles' Anschauungen zur Entstehung und Funktionsweise von Leben..* Stuttgart: Franz Steiner Verlag.

Liu, Y. et al., 2006. Geometric Modelling with Conical Meshes and Developable Surfaces. *ACM TRANS. GRAPHICS,* pp. 681-689.

Maki, F., 1964. *Investigations in Collective Form,* St. Louis: Washinton University.

Maki, F. G. J., 1962. *Linkage in Collective Form,* St. Louis: Washington University Press.

Markus, T. A., 1969. The Role of Building Performance Measurment and Appraisal in Design Method. *Design Methods in Architecture.*

Maver, E. D., 1970. Appraisal in the Building Design Process. *Emerging Methods in Environmental Design and Planning.*

Mayne, T., 2011. *Combinatory Urbanism: the Complex Behaviour of Collective Form.* Culver City, CA: Stray Dog Cafe/ Morphosis Architects.

McCabe, T. J., 1976. A Complexity Measure. *IEEE Transaction on Software Engineering*, December, pp. 308-320.

McCaskill, J., 2012-2015. *MICREAgents Project.* [Online]
Available at: http://www.micreagents.eu//index.html
[Accessed 01 12 2017].

McNeel, R. a. A., 2017. *Rhinoceros.* [Online]
Available at: https://www.rhino3d.com/
[Accessed 01 12 2017].

Meyer, B., 1997. *Object-Oriented Software Construction.* 2nd ed. California: Prentice Hall.

Mignonneau, L. & Sommerer, C., 2008. The Art and Science of Interface and Interaction Design. In: C. Sommerer, L. C. Jain & L. Mignonneau, eds. *Media Facades as Architectural Interfaces.* Berlin: Springer, pp. 93-104.

Mitchell, W. J., 1990. *The Logic of Architecture.* Cambridge, Massachusetts: MIT Press.

Noboru, K., 1960. *Metabolism/1960.* Tokyo, Bijutsu shuppansha.

Oshima, K. T., Ibanez, D. & Katsikis, N., 2014. On Metabolism and the Metabolists. *New Geographies,* pp. 098-107.

Phylosophy, S. E. o., 2015. *Standford Encyclopedia of Phylosophy.* [Online]
Available at: https://plato.stanford.edu/entries/architecture/
[Accessed 23 2 2018].

Raynolds, C., 2001. *Boids.* [Online]
Available at: http://www.red3d.com/cwr/boids/
[Accessed 02 19 2017].

Reeves, S., Banford, S. & O'Malley, C., 2013. *Designing Interfaces in Public Settings.* [Online]
Available at: https://www.blasttheory.co.uk/wp-content/uploads/2013/02/research_designing_interfaces_for_public_places.pdf
[Accessed 12 4 2017].

Roland, C., 1970. *Frei Otto: Tension Structures.* Westport, Connecticut: Praeger Publishers.

Rosenberg, D., 2009. *Designing for Uncertainty: a Novel Shapes and Behaviours using Scissor-Pair Transformable Structures..* Cambridge: MIT Master of Science Thesis.

Sasaki, M., 2005. *Flux Structure.* Tokyo: TOTO.

Schon, D. A., 1983. *The Reflective Practitioner: How Professionals Think in Action.* New York: Basic Books.

Seitinger, S., Perry, D. & Mitchell, W., 2009. Urban Pixels: Painting the City with Light. *CHI,* pp. 839-848.

Sevtsuk, A. & Kavlo, R., 2014. A Freeform Surface Fabrication Method with 2D Cutting. *SIM AUD,* 46(7).

Sterk, T., 2003. Building Upon Negroponte: a Hybridized Model of Control Suitable for Responsive Architecture. *eCAADe,* Volume 21, pp. 407-414.

Thomson, D., 1961 (First ed. 1917). *On Growth and Form.* Cambridge, United Kingdom: Cambridge University Press.

Tomitcsh, L. G. T., Vande, A. M. & Renan, S., 2010. Information Sky: Exploring and Visualising Information on Architectural Ceilings. *DOI,* Volume 10, pp. 100-105.

Warren, J., 2003. *Unencombered Full Body Interaction In Videogames.* s.l.:Parson's School of Design.

Weiser, M., 1991. The Computer for the 21st Century. *Scientific American,* 25 Feb, 22(2), pp. 60-63.

Wiberg, M., 2015. Interaction Design Meets Architectural Thinking. *Interactions,* 25 Feb, 22(2), pp. 60-63.

Williams, N. H., 2017. *Plugin Practice: Recasting Modularity for Practice.* Melbourne: RMIT University.

Yaglom, I. M., 1962. *Geometric Transformations.* book 8 ed. Washington D.C.: Mathematical Association of America.

Zhongjie, L., 2010. *Kenzo Tange and the Metabolist Movement.* NY: Routledge.

Zuk, W. & Clark, R. H., 1970. *Kinetic Architecture.* New York: Van Nostrand Reinhold.

List of Illustration

Figure 17 Robert Cameron

Figure 18 Andrei Smolik & Robert Cameron

Figure 19 Andrei Smolik & Robert Cameron

Figure 20 Andrei Smolik

Figure 21 Michaela Photography

Figure 22 Andrei Smolik

Figure 23 Andrei Smolik

Figure 24 Andrei Smolik

Figure 25 Andrei Smolik

Figure 26 Andrei Smolik

Figure 27 Andrei Smolik

Figure 28 Andrei Smolik

Figure 29 Teng Weng Chen, The Idea Factory

Figure 30 Andrei Smolik

Figure 31 Andrei Smolik

Appendix

A. Publications

International Conference Proceedings:

Smolik, A., Chang, T., Datta, S. 2017, '*Prototyping Responsive Carrier-Component Envelopes'*, CAADRIA, Chaozhou, China.

Smolik, A., Chang, T., Datta, S 2016, *'Responsive Interaction in Dynamic Envelopes with Mesh Tessellation'*, CAADENCE, Budapest, Hungary.

Magazine Article:

Smolik A., Cameron, R. 2017. 'cringeMACHINE' in Cockton, G., Barbosa, S. *Interactions.* 2017 May-June: 26(3): 8-9. DOI: 10.1145/3095803

B. Full articles

Responsive Interaction in Dynamic Envelopes with Mesh Tessellation

Sambit Datta, Smolik Andrei, Tengwen Chang

Curtin University
Australia
National Yunlin University of Science and Technology
Taiwan

e-mail: Sambit.Datta@curtin.edu.au, andrei.smolik1@curtin.edu.au, tengwen@yuntech.edu.tw

Abstract: The paper will report on an ongoing experimental research collaboration to develop a conceptual architectural envelope that responds to feedback from external stimuli. The paper builds on our research on the construction of interactive envelope components based on the manipulation of mesh tessellations and the development of responsive interaction with architectural elements. In the context of our research, chameleon ceiling looks at ways to integrate various systems (geometric, structural and electronic-responsive) and their effects on modular, component-driven systems construction through computation and digital fabrication.

Keywords: Dynamic envelopes, mesh tessellation, sensor interaction, interactive architecture, digital fabrication.

1 Introduction

The objectives of the first phase of this investigation are to integrate geometry, material and fabrication with interactive/responsive mechanisms in IoT research. The end goal is to develop a full scale responsive ceiling envelope for YunTech Digital Media Design Centre to provide a framework for experimentation with ambient interactivity. The conception and construction of the prototypes take advantage of parametric and digital fabrication strategies for material exploration, responsive interaction logic as well as developments in microelectronics.

We discuss the theoretical framework that we used to conceptualise the design, some prototypes done in both labs, implications of geometry on HCI and space and the way in which HCI can be reconceptualised as an architectural problem where there are important correlations between the design of computing systems and the design of physical spaces and places.

2 Theoretical Framework

2.1 Ubiquitous Computing

From the point of view of human computer interaction (HCI), architectural space and living settings in general present an opportunity for expanded possibilities of interactive environments. The concept of a 'disappearing computer' predicts that with the advancements of digital technologies and their increased availability, individual devices and appliances would give way to ubiquitous systems that would be incorporated into all facets of our everyday lives [1]. Embedded computation has, since Weiser's predictions expanded to the fields of art and design with problems that often cross disciplinary boundaries and present us with what some call diffuse problems [2]. These are complex areas of inquiry that require interdisciplinary approaches. In order to develop design strategies for the interactive envelope, we draw on the previous taxonomies of ambient display information systems [3] as well as theory and research on architectural elements as an ambient and a story telling medium.

2.2 Ceilings as story-telling medium

We ground the design framework on ceilings as a potential platform for new ways of visualising information and creating ambient displays. From cave paintings to Egyptian temples to Medieval and Renaissance periods, the ceiling was used as a storytelling medium. Through time, people looked up to increase their knowledge of their environment and the world around them making it a natural opportunity for a responsive envelope [4]. Today we are more reliant on digital appliances and hand-held devices, relegating architecture to the background. Tectonically, most contemporary ceilings consist of modular panels suspended from building structure concealing ducts and wiring with the open plan increasingly being co-opted as an efficient means of achieving maximum density especially in office environments [5]. This creates an effect of a large homogenous surfaces interrupted by AC vents, sprinklers and other service outlets. Architectural ceiling has moved away from being a storytelling medium conveying abstract information to purely a functional feature. Chameleon ceiling attempts to reconceptualise ceilings as ambient story-telling envelopes by incorporating ubiquitous computing, ambient information visualisation and component driven digital design and fabrication.

2.3 Architectural Component Design

The combination of architectural parametric design with interactive and adaptive control systems is a relatively new area of inquiry with many architectural schools initialising new programs that are dealing with material and computational research [3]. This project opens up the possibility for connecting physical architectural elements to be deployed as means of testing various aspects of HCI through an abstract cellular sensing and communication.

The *Chameleon Ceiling* research is centred on the hypothesis that a deeper understanding of spatial interactive systems and ubiquitous computing can be achieved by exploring the possibilities of three-dimensional cellular envelopes as installation prototypes to test out various systems of interaction within architectural space. Computational design is an experiment which investigates behaviour of increasingly complex systems both physical/architectural and interactive [4]. The following text describes the experimental studies through which we are exploring this hypothesis.

2 Prototypes

2.1 Carrier components and digital fabrication

To develop digital fabrication and basic computational geometric techniques in our lab, we develop a post-graduate curriculum, prototyping simple digital fabrication methods centring on carrier components and free-form surfaces (Figure 1). Students are asked to create a parametric prototype based on a variety of design scenarios while considering material selection and fabrication methodology. Through this process we keep on pushing available digital fabrication methods in order to apply them in a more robust manner to a greater variety of materials using a larger sample of geometries in design.

Figure 1: Postgraduate work developing digital fabrication methods

2.2 Components as pixels

The design concept was developed through post-graduate research focusing on aggregative form and construction of an interactive prototype (Fig 2). Interactive installations deploying high-resolutions tend to be non-ubiquitous, flat, bounded and rely on centralized control systems [5]. The base idea formed from looking at geometries and compositional approaches that begin from a series of simple rules and form complex structures that are incomplete or have the ability to renew and

grow where *'the element suggests a manner of growth, and that, in turn, demands further development of the elements, in a kind of feedback process'* [6].
The components acted as spatial pixels and carried LEDs and were activated through one centrally located microphone sensor. They can be switched or modified, independent of their position or internal structure. Strategically deploying low-resolution spatial ambient light makes context surroundings more legible [5].

The form was based on a simple script that added a truncated tetrahedron to the proceeding shape creating additive, crystalline-like geometries from repeating elements. These strategies are derived from structures such as Weaire-Phelan and are present in both the soap bubbles and polycrystalline solids [7]. Unlike Weaire-Phelan however, this approach subdivides space quasiperiodically and thus it is difficult to articulate in such a way as to enclose or be structurally complete. For the next step we used a different geometric approach, focusing on much simpler generation of a continuous carrier surface with a repeating component tessellating it. We did, however retain the idea of components as pixels that act as independent cells on a tessellating grid.

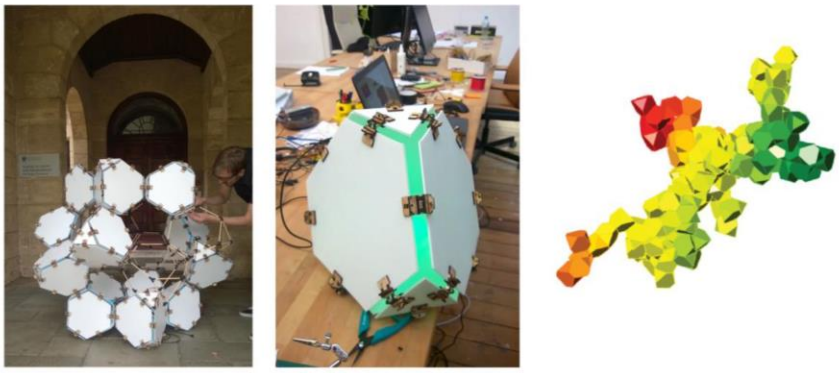


Figure 2: Postgraduate work incorporating aggregative components and sensor technology

2.3 Interactive ceiling

2.3.1 Geometry

Development of the *chameleon ceiling* prototype began by establishing an envelope that acts as a carrier surface for repeating sets of components forming a pixel grid. The following design iterations began testing spatial and architectural strategies for positioning the pixel envelope between the main entrance and the adjacent wall spanning the length of the staircase connecting ground floor to upper mezzanine level inside the YunTech's interactive design lab (Fig. 3).

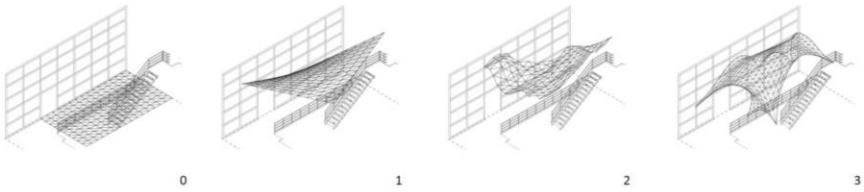


Figure 3: Design iterations of an 'architectural pixel envelop

Initial design was a simple hyperbolic paraboloid with triangular tessellation. Second iteration was used in prototyping of scaled components and virtual interaction scenarios and is a free-form NURBS surface that acted as a carrier for triangular tapered extrusion cladding elements (Fig 4). The third iteration is physically simulated mesh which forms a vaulted structure. These design iterations were done using Grasshopper [9] and Kangaroo plugin [10].

The major constraints of the envelop surfaces are structural. In Figure 3(2), the freeform surface is envisioned to be supported from the ceiling by a tertiary system of proprietary steel ties that connects the structure of the installation to the ceiling.

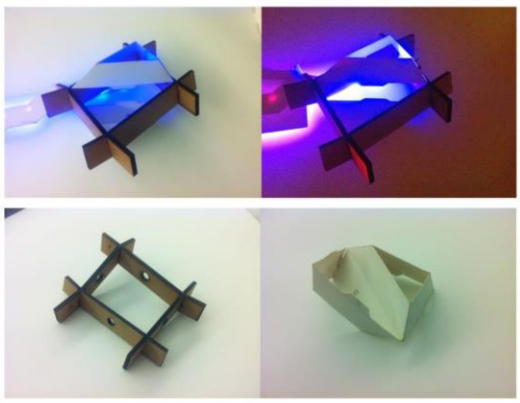


Figure 4: Testing of component systems (scale 1:20)

This means that components would require a secondary structure –a skeleton which holds component in place providing a rigid framework to connect to elements of the room with tertiary structure. Figure 3(3) attempts to minimise tertiary structure by implementing a physics based geometry solver to put

components mostly under compression, creating a catenary canopy system. The geometry of individual pixel components also begins to fluctuate depending on variation in curvature and stresses- some becoming much larger and stretched near anchor points with additional surface areas.

Another impact is surface resolution. Given the site area of approximately 8mx4.5mx5m, variation in component density and thus, component scale, has impacts on both the ambient display resolution which in turn affects the complexity of structure and construction.

For the purposes of scaled prototypes and scenario testing, we established a one-to-one relationship of pixel to component. Effects of distortion of individual components and non-orthogonal mesh edge alignment as well as methods of fabrication of multiple, unique structural components and claddings are still under investigation.

2.3.2 Interaction

The project also entails a parallel body of research into combinatory approaches to computer simulation of interaction (Fig 5). Using simple data sets and attractors, these were designed to refer to potential ambient data displays and traction of individuals through space. This framework allowed us to simulate simple interactivity scenarios and their affects. Further experimentation was done using Firefly plugin for Grasshopper [11]. This allowed us to link the camera inputs directly to the geometry and more closely approximate interaction-to-geometry affects in the component based design. The firefly generates an undulating mesh as a visual representation of camera bitmap brightness from which we sample Z-coordinates related to the components of the installation. We than use these parameters to create an RGB grid of pixels. Depending on how these are chosen to be mapped to the components, there is a large number of possibilities depending on how colour parameters are used. In this particular test, there is only one parameter for each pixel, however it is possible to shift values to generate multi-coloured scenarios or use other information to map to different colour values entirely.

The low resolution quality of the mesh creates opportunities and constraints for both ambient qualities of interaction (with feedback of movement through space) that could retain or aggregate colour intensities as more people are travelling through to more specific information that could be mapped to different parts of the envelope. Individuals could learn to adapt to different methods of representation and information communication [12]. This presents interaction design with a rich platform for experimentation and testing of various combinations of input/output scenarios.

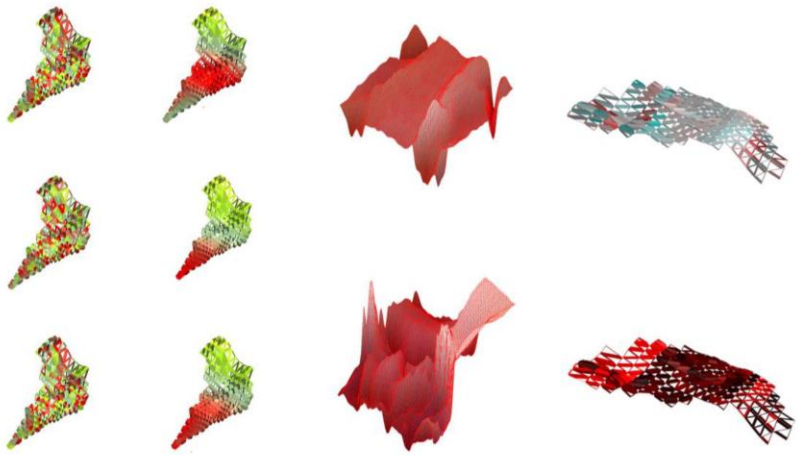


Figure 4: Simulating LED interaction in geometry

3 Discussion

The first stage of explorations into architectural dynamic ceiling envelopes has given us some insight into research issues associated with ambient architectural displays. These present an interplay between physical structural systems and systems of interaction.

In terms of geometry and fabrication, additional techniques have to be developed to minimise variation in component geometry to streamline fabrication. This could incorporate flexible materials in physical assembly or new mesh optimisation techniques in modelling. The structural forces need to be engineered from the geometry of the ceiling itself to relieve it of redundant structural features.

Interaction needs to be further prototyped on clusters of components to test intensity and legibility of light, assembly sequences and optimisation for interchangeable output mechanisms. In the next stage interaction will also be tested at full scale on site.

4 Conclusion

Technology is rapidly advancing and seeping into all areas of contemporary life. As HUI and ubiquitous computing research expands to include larger elements of human environment and interaction, we will see traditional architectural elements transformed. The initial research phase presented here attempts to examine architectural geometry in light of responsive design and explore potentials of cellular interactive envelopes. In the next phase of work we plan to construct full

scale prototypes and begin testing structural and interactive properties of *chameleon ceiling* on site. We expect that the results from these experiments will provide further opportunities for physical refinement and bring it closer to full scale application of ambient interaction.

References

[1] Weiser, M., The Computer for the 21st Century, *Scientific America*, 265(3), 1991, p. 94-104.

[2] Dade-Robertson, M., Architectural User Interfaces: Themes, Trends and Directions in the Evolution of Architectural Design and Human Computer Interaction, *IJAC*, 11(1), 2013, p. 1-19.

[3] Tomitsh, M., Kappel, K., Lehner, A., Grechenig, T., Towards a Taxonomy for Ambient Information Systems, *Pervasive 2007 W9 Ambient Information Systems* (May 13, 2007).

[4] Tomitsch, M., Greechening, T., Vande More A. and Renan, S., Information Sky: Exploring and Visualising Information on Architectural Ceilings, *DOI'10*, 2010, p. 100-105.

[5] Meagher, M., Huang, J. and Gerber D., Revisiting the Open Plan: Ceilings and Furniture as Display Surfaces for Building Information, *The Proceedings of International Conference Information Visualisation*, IV'07, 2007, p. 601-606

[6] Seitinger, S., Perry, D., and Mitchell, W., Urban Pixels: Painting the City with Light, *CHI '09*, 2009, 839-848.

[7] Velikov, K., Thün, G., Ripley, C., Thick Air, *Journal of Architectural Education (JAE),* 2012, p. 69-97.

[8]Gengnagel, C., Kilian, A., Palz, N., Scheurer, F., *Computational Design Modelling: Proceedings of the Design Modelling Symposium Berlin 2011*, Springer: Berlin, 2011, p. VIII.

[9] Maki, F., *Investigations in Collective Form*, Washington University, 1964, p. 19.

[10] Wearie, D., Hutzler, S., *The Physics of Foams,* Oxford: Clarendon Press, 1999.

[11] http://www.grasshopper3d.com/(4-29-2016)

[12] http://www.grasshopper3d.com/group/kangaroo (5-3-2016)

[13] http://www.fireflyexperiments.com/ (4-29-2016)

[14] Wisneski, C., Ishii, H., Dashley, A., Gobrbet, M.m Brave, S., Ullmer, B. and Yarin, P., Ambient Displays: Turning Architectural Space into an Interface between People and Digital Information, *CoBuild, 98*, Berlin, Springer, 1998, p. 22-42 (23),

P. Janssen, P. Loh, A. Raonic, M. A. Schnabel (eds.), *Protocols, Flows and Glitches, Proceedings of the 22nd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2017*, 521-529.

PROTOTYPING RESPONSIVE CARRIER-COMPONENT ENVELOPES

ANDREI SMOLIK[1], TENGWEN CHANG[2] and SAMBIT DATTA[3]
[1,3]*Curtin University, Australia*
[1,3]*{Andrei.Smolik1|Sambit.Datta}@curtin.edu.au*
[2]*National Yunlin University of Science and Technology, Taiwan*
[2]*tengwen@gemail.yuntech.edu.tw*

Abstract. The capacity to respond dynamically to changes in external and internal environments open new possibilities in the interaction between buildings, humans and the environment. The development of dynamic envelopes requires the integration of various systems- geometric, structural, and electronic-responsive and their interaction. The paper reports the results from the "Dynamic Cloud Project" and presents a design and fabrication methodology to integrate kinetic behaviour with material constraints; the simulation of responses by connecting components with programmable input and behaviour. The paper presents a modular, component-driven systems construction based on a carrier-component surface geometry called responsive carrier-component envelope (RCCE) and describes the modelling, fabrication and assembly of such envelopes. The protocols developed in the project are reported in the paper and highlight the opportunities and consequences of how local components relate to the whole carrier envelope with multiple constraints and scale considerations. The results of the prototyping and experimentation with this project are reported in the paper. The paper also discusses future applications of the research and outlines new possibilities and design opportunities in prototyping responsive carrier-component envelopes.

Keywords. Dynamic envelope; carrier component mesh; sensor interaction; interactive architecture; digital fabrication.

1. Introduction

The research is inspired by formative ideas proposed in the sixties and seventies which speculated on the possibility of an indeterminate architecture responsive to uncertainty, incompleteness and emergent states. Urban utopian proposals by Archigram and Japanese Metabolists (Cook 1990; Maki 1964) speculated on a shifting assembly of parts that make up a whole while kinetic architecture of

William Zuk and Rogert H. Clark conceived of in the seventies envisioned moving components and entire structures adapting to the needs of its users (Zuk & Clark 1970). Nicholas Negroponte, a forerunner of the concept of responsive architecture, in 1970's built a body of work attempting to define and produce it by integrating computation into spaces and structures through recognition, intention, contextual variation and meaningful computed responses (Sterk 2003).

The two common themes that stem from these approaches is the ability of design to adapt to a variety of states akin to a real-time parametric model, changing the morphology and architectural understanding of part-to-whole relationships and enabling user's input and emergent situations to influence these states- achieving a more direct relationship between architecture, its occupants and their needs. In this sense, architectural design is invoked as an integrated spatial strategy with robotic protocols inseparable from the architectural design and its intent.

Today, these technologies are at the forefront of architectural research and can be broadly divided into two categories: digital fabrication/assembly and dynamic architecture. While the former has a clear agenda and a well-defined strategy as new means of production and communication of a digital design process to final outcome, the latter still remains somewhat elusive (Rosenberg 2009). While advances and accessibility of certain technologies in recent years allowed architects to experiment with ideas of interaction and kinetic actuation more directly, the larger question of synthesis- how all the disparate areas of knowledge and systems can be integrated into approaches that can be applied more directly to larger architectural structures remains unclear.

In this paper we look at the possibility and challenges of structural integration of dynamic envelopes. By looking at the example of the Dynamic Cloud project, we propose a design criteria which focuses on the interaction between part and a whole- how structure, interaction, materiality and geometry is impacted by assembling "smart" or complex discrete components into larger, potentially indeterminate wholes- what challenges emerge and how we may be able to systematically control these in the future.

2. Towards responsive carrier component envelopes

Computing technology is rapidly advancing and seeping into all areas of contemporary life and architecture. Responsive surfaces allow for interactive communication between people and their environment through the use of micro-controllers, sensors and actuators. Responsive envelopes have been classified into three broad categories in the literature, namely media facades (Mignonneau & Christa 2008) dynamic envelopes (Tomitsch et al. 2010) and interactive systems (Seitinger et al. 2009). This paper presents the concept of a responsive carrier-component envelope (hereafter RCCE), a physical prototype that dynamically integrates, environmental information in real time, and responds by changing its geometric state. The following features are used to define the design criteria for the development of the RCCE prototype:

1. The structural support grid and overall geometry should be defined by a carrier surface geometry. The carrier propose a self-supporting, integrated structural grid

geometry, avoiding peripheral structural or cladding elements;
2. The carrier surface should be decomposable into an aggregation of kinetic components. The kinetic elements are aggregated to minimise cost and complexity and provide a coherent integration between responsive and physical/structural components; and
3. Each discrete component should support open ended, reprogrammable sensor input-output with the environment. Components should be reprogrammable and re-configurable to allow a broader range of experiments with interactivity. The network should support accessibility and flexible placement of sensor infrastructure.

Our end goal is to make a self-supporting surface geometry using reciprocal kinetic components that operate on computation protocols and allow it to exhibit self-organisational properties. The Dynamic Cloud project brief was to design and prototype an RCCE for display and installation at the Shenzhen Maker Faire held in November 2016. The project was undertaken in collaboration between IdeaFactory (YunTech) and CodeLab (Curtin University) under the supervison of Carl Yu (OneWork.io), a startup company with expertise on Internet of Things (IoT). The aim of the project was to demonstrate an RCCE prototype using design-oriented fabrication approach. Design and research were conducted on multiple aspects of the process, remotely communicating between the teams in Shenzhen, Taiwan and Perth, Australia. The Curtin team focused on the design of the carrier surface and kinetic component prototyping while The YunTech team focused on the fabrication and making of the RCCE and put together prototypes, improved aspects of geometry and code and took part in the final assembly.

3. Theory Application

The core of RCCE is the assembly of larger structural surfaces from a collection of self-contained autonomous kinetic components that respond to sensor based interaction.

4. Carrier Geometry

Grid shells have been studied and applied to full scale built architecture by many practitioners and scholars. The carrier geometry of the RCCE was initially conceived of as a self-supporting parabolic grid shell that consists of multiple systems that allow (potentially) indeterminate bracing and structural support, kinetic actuation and communication and cladding layer which could expand the possibilities of what tasks the carrier component envelopes are asked to perform. The structural complexity of the carrier surface and lack of time meant that the parabolic structure was not pursued beyond the prototyping phase for the final installation. For tessellation of the geometry of the dynamic cloud project we used a simple surface triangulation that follows a hexagonal pattern (figure 1). The component nature of conical meshes on which global geometry is based creates opportunities for bottom-up approaches to component-based envelopes but presents structural challenges in relation to stresses and connections at vertices (Liu et al. 2006).

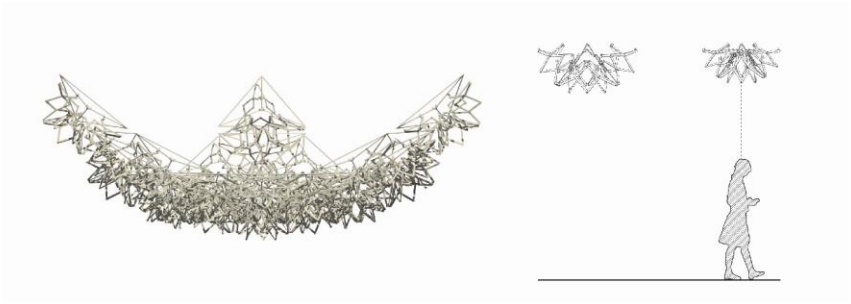


Figure 1. Prototype responsive ceiling design with kinetic component.

5. Kinetic Component Design

The self-organisation is constrained by surface geometry on which the component is aggregated. Parts have a degree of freedom in a complete system and work independently while responding to their neighbors effecting changes on the larger scales. This allows for a less deterministic part-to-whole relationships to emerge from interactions of local protocols and a multitude of external stimuli. In the current prototype, components in RCCE are comprised of scissor truss assembly based on Hoberman sphere (Hoberman 1990) guided by sliding arms and allowing a single degree of freedom in a symmetrically polar array. The components follow rules of local independence and global correspondence to form iterative responses and spatial configurations based on various local inputs (figure 2).

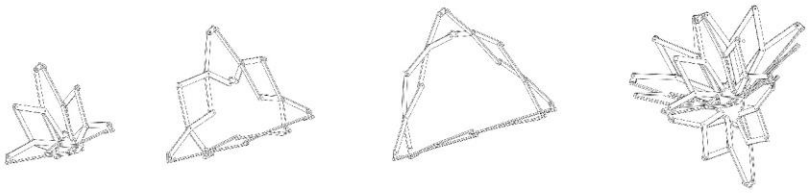


Figure 2. Early component configuration prototype models for kinetic components based on the scissor truss.

6. Sensor Interaction

Each component is an autonomous mechanism with an ultrasonic sensor (HC-SR04) driven by Arduino microcontroller. The sensor detects proximity of objects in front of it with a maximum range of 5 meters. Upon detection, it activates a geared motor connected to the kinetic truss through linear motion and begins contracting it. Once the sensor stops detecting or the collision switch is activated when

the truss is fully collapsed, the script pauses and proceeds to return the mechanism to its extended configuration.

7. Simulation and Prototyping

The RCCE is prototyped with top-down and bottom-up approaches using modelling and scripting techniques to get information about geometry and kinetic properties of the prototype. In the bottom-up approach, the scissor truss assembly derived from four sides of a dodecahedron is used to generate geometry in Grasshopper and simulate its movement using a spring system in Kangaroo (figure 3). The complexity of the assembly and its kinetic actuation, variation of parameters and scale are tested.

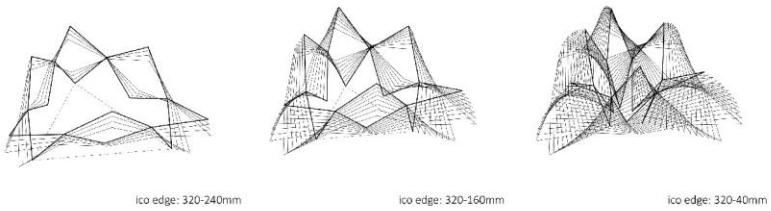


Figure 3. Simulation of component motion.

The top-down approach involved creation of a grid shell surface structure for carrier components. One of the challenges with introduction of a curved structural grid is variation in the geometry of parts. This presents significant challenges in terms of assembly of complex interactive components. Originally, the structural grid fabrication was based on the approach presented by Andres Sevtsuk and Raul Kavlo in the SUTD pavilion (Sevtsuk & Kavlo 2014). However, this approach was replaced by a simpler carrier surface. The next step was to standardize kinetic component parts, namely the guiding rods to fit the curving surface. A standardization script was developed (figure 4) that fitted equilateral triangles using geometric transformation per face (Yaglom 1962). If the face is too small, the solution is culled. However, this causes several problems- orientation of components is misaligned with the original diagrid geometry, also the connection of the rod to the outside support structure generated variable insertion angles that proved intractable in the fabrication process. For the Shenzhen fair we settled on planar surface tessellation with equilaterals and separated the structure from the kinetic component grid. This design compromise simplified the kinetic component and enabled the delivery of local interactions without structural ramifications.

8. Material Assembly

The basic unit is built on a triangle containing tetrahedral scissor truss geometry (figure 4). Structurally, the components are made of three main parts: the struc-

tural frame that houses all the parts and connects it to its neighbors (1), guiding rail assembly which contains all sensing, actuating and logic components (2) and kinetic scissor truss assembly (3). 34:1 motor connects to three linear belts that are guiding the base points of the truss. There is a binary switch that communicates to the script terminal position of the truss under maximum contraction.

A flexible multi-axis 3d-printed connectors allow us to connect the kinetic scissor truss to a planar rail frame which can be animated by linear motion. Early prototypes contained springs which kept component in tension while the final prototype used timing belts and motors to control position more precisely and eliminated kinetic energy which made motion more unstable given relatively large number of parts per truss. The design uses a number of standard industrial components like steel shafts and connector brackets to reduce construction time and cost (figure 4). A few 3d printed components were used to connect assembly at key intersections with unique angles. In this case, 3d printing is used as part of the combinatorial approach to construction where many different materials and parts come together as oppose to creating complex, continuous forms.

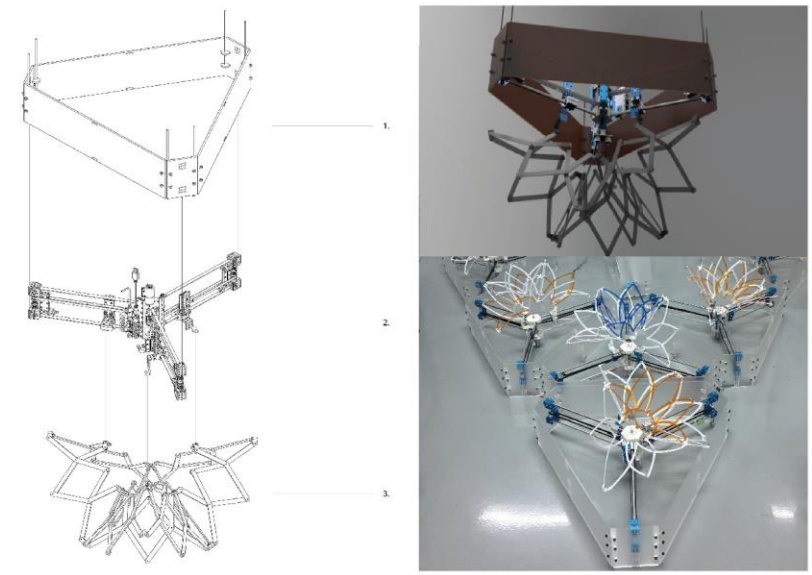


Figure 4. Final component assembly (left) and Design variations of 3d printed joints and connectors (right).

9. Control System

Control system uses standard Arduino IDE written in C for individual components with the controller script being uploaded to local microcontroller boards mounted on the structural frame. The microcontroller is connected to the motor, ultrasonic sensor, an LED strip and a switch to determine position of the kinetic truss. This

allows components to 'sense' presence and proximity and respond by retraction or contraction. The precise timing allows for real-time control and synchronization of motor positions in relation to external stimuli.

10. Exhibition Prototype

Over the period of three weeks, the teams communicated and exchanged 3d models, scripts and other documentation relating to prototype construction. The Curtin team assembled design documentation while Yen Tech team prototyped various design iterations using digital fabrication equipment at their facilities before proceeding to install the prototype on site in the Shenzhen MakerFaire. The final 1:1 scale prototype was realized with three days of on-site construction made out of 16 fully-assembled and functional components which were fixed to a stable temporary structure with steel cable rods (figure 5). These arrangements provided the right constraints for the model surface to be suspended over temporary public space allowing for minimal lateral movement and interference between components. Components performed interaction with no identifiable failures over the course of three days.

Figure 5. RCCE final installation and interaction at Shenzhen Maker Faire, November 2016.

11. Conclusion

The results of the prototype experiments for the development of responsive carrier component envelopes are:

1. the development of the carrier surface and its subsequent articulation with components requires geometric resolution at two scales. The tessellation of the carrier surface must be rationalized to account for both part-whole relationships as well as control of component variation and scaling.
2. the digital to physical translation from rationalized geometry to material requires further inputs in the selection of materials, their tectonic properties as well as a consideration of their assembly; and
3. the incorporation of responsive elements in the form of electronic components and sensors requires careful consideration of the interaction logic, input-output behaviours as well as sizing and resolution of components. An important element in interaction design is the trade-off in resolution- density of components corresponding to the component positioning. Denser iterations produce greater resolution but create more complexity in terms of set up and fabrication as well

as driving up costs.

The RCCE is an experiment in architectural design that gives us insight into construction of complex architectural assemblies incorporating digital fabrication, dynamic forms, system design and planning. RCCE is work in progress and due to its complexity and wide range of interdisciplinary knowledge required, will require expended collaboration in the future. The main problems encountered are structural- control required to script and test a largely bottom-up system in terms of its structural dynamics and force distribution is considerable. Although the behavior is emergent- the form still remains static. This problem will be addressed in further design iterations to control the local parameters influencing the surface tessellation and have more control over aggregation of local geometry into a larger whole. This could be expanded upon by looking at the way components connect, degrees of freedom allowed per component and the way cladding layer can be arranged more continuously rather than a collection of disparate kinetic scissor truss assemblies. The future implementation will include communication protocol between components to allow for coordinated transformation of the structure.

Acknowledgements

The authors acknowledge the contributions of the team leader of the project "Dynamic Cloud" Carl Yu (onework.io) and the following students: Rex Auyeung (Curtin University); Tsai-Tsai Hsieh, Arex Lee, Yun Ru Chen, Chen Junyan, Wayne Lin, Monica Shen, Sherry Huang (Idea Factory, Yun Tech).

References

Cook, P.: 1990 (ed.), *Archigram*, Princeton Architectural Press, New York.

Liu, Y., Pottman, H., Wellner, J., Yang, Y.-L. and Wang, W.: 2006, Geometric Modelling with Conical Meshes and Developable Surfaces, *ACM TRANS. GRAPHICS*, **25**, 681–689.

Maki, F.: 1964, *Investigations in Collective Form*, Washington University.

Mignonneau, L. and Christa, S. 2008, The Art and Science of Interface and Interaction Design, *in* L. Mignonneau and S. Christa (eds.), *Media facades as architectural interfaces*, Springer, Berlin Heidelberg, 93-10.

Rosenberg, D.: 2009, *Designing for Uncertainty: Novel Shapes and Behaviours using Scissor-Pair Transformable structures*, Master's Thesis, MIT.

Seitinger, S., Perry, D. and Mitchell, W.: 2010, Urban Pixels: Painting the City with Light, *CHI*, **09**, 839-848.

Sevtsuk, A. and Kavlo, R.: 2014, A Freeform Surface Fabrication Method with 2D Cutting, *Proceedings of the SIM AUD, 2014*.

Sterk, T.: 2003, Building Upon Negroponte: A Hybridized Model of Control Suitable for Responsive Architecture, *eCAADe*, **21st**, 407-414.

Tomitsch, M., Greechening, T., Vande More, A. and Renan, S.: 2008, Information Sky: Exploring and Visualising Information on Architectural Ceilings, *no title given*, **10**, 100-105.

Yaglom, I. M.: 1962, *Geometric Transformations (book 8)*, Mathematical Assn of Amer, Washington D.C.:.

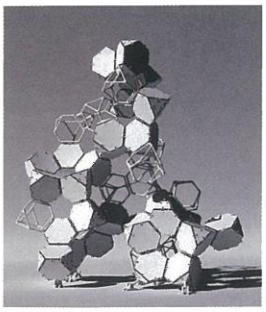
Zuk, W. and Clark, R.: 1970, *Kinetic Architecture*, Van Nostrand Reinhold, New York.

HOW WAS IT MADE?

Specs
Materials: 3mm plywood, 3mm twin-wall polypropylene sheet, timber dowels, ABS plastic, 36mm LED pixels, microphone, Arduino Uno with Waveshield
Tools: Laser cutter, CNC cutter, desktop FDM 3D printer, Rhino3D + Grasshopper, Processing, nuts and bolts, cable ties

cringeMACHINE

Describe what you made.
cringeMachine is a temporary installation composed of 20 truncated tetrahedrons that respond to the sounds around it. Our initial idea was to create a transformable sculpture made of self-similar parts that could be assembled into a variety of configurations. When you make sufficient noise around it, the cringeMachine sends a pulse of lights through a sequence of LEDs built into its component parts, varying in color and intensity.



→ Computer rendering of original design concept.

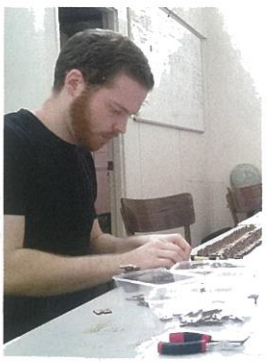
What materials and tools did you use?
We used thin twin-wall polypropylene sheets for solid LED component cladding, laser-cut plywood for the flexible hinge connectors, pine wooden dowels, and 3D-printed pronged fittings for the hollow structural frame. The use of expensive-looking (but in reality, dirt-cheap) white plastic in combination with natural timber tones created an interesting visual blend of materials reflecting the combination of precisely machined and meticulously hand-crafted construction techniques.

Did anything go wrong?
Halfway through the process, we realized that using hinged friction joints was not going to be structurally viable while also maintaining the modularity of the structure at the desired scale. The way we got around it was by fixing hinged joints into position and then settling on the most stable configuration of parts: a truncated dodecahedron. This is interesting since some naturally occurring tessellations (like crystals) tend to create forms that are similarly stable.

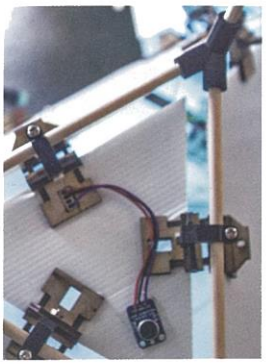
What was the biggest surprise in making this?
It was interesting to discover that even making simple interactive systems could elicit great interest and response—the sculpture became more and more like a reflector of the performers in the space surrounding it rather than a singular sculptural statement. Its reaction to sound often became obfuscated, appearing more like a flickering candle with a mind and thoughts of its own. The interaction felt natural and less mechanical than what we were initially expecting.

The interaction felt natural and less mechanical than what we were initially expecting.

How would you improve on this if you were to make it again?
In terms of form, we would definitely redesign the friction hinge connectors; though their complexity added to the overall aesthetic effect, they potentially could have been simplified, making them easier to fabricate.

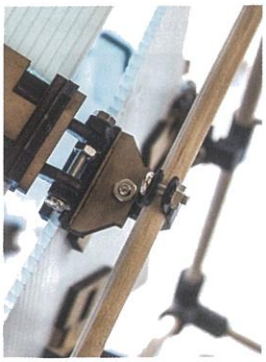


→ Assembling flexible hinge joints.



→ Detail of cell joint and sound sensor.

→ Assembling the ring module.



→ Detail of hollow-cell and solid-cell joint.

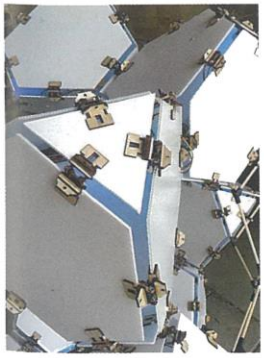
Interlocking them mechanically would also increase overall stability and potentially fulfill our initial aspiration of making the structure modular. We would also like to expand on the ideas of the modularity of responsiveness and structural modularity—making components independently addressable as well as potentially building awareness and simple behaviors into individual cells.

Andrei Smolik,
Curtin University

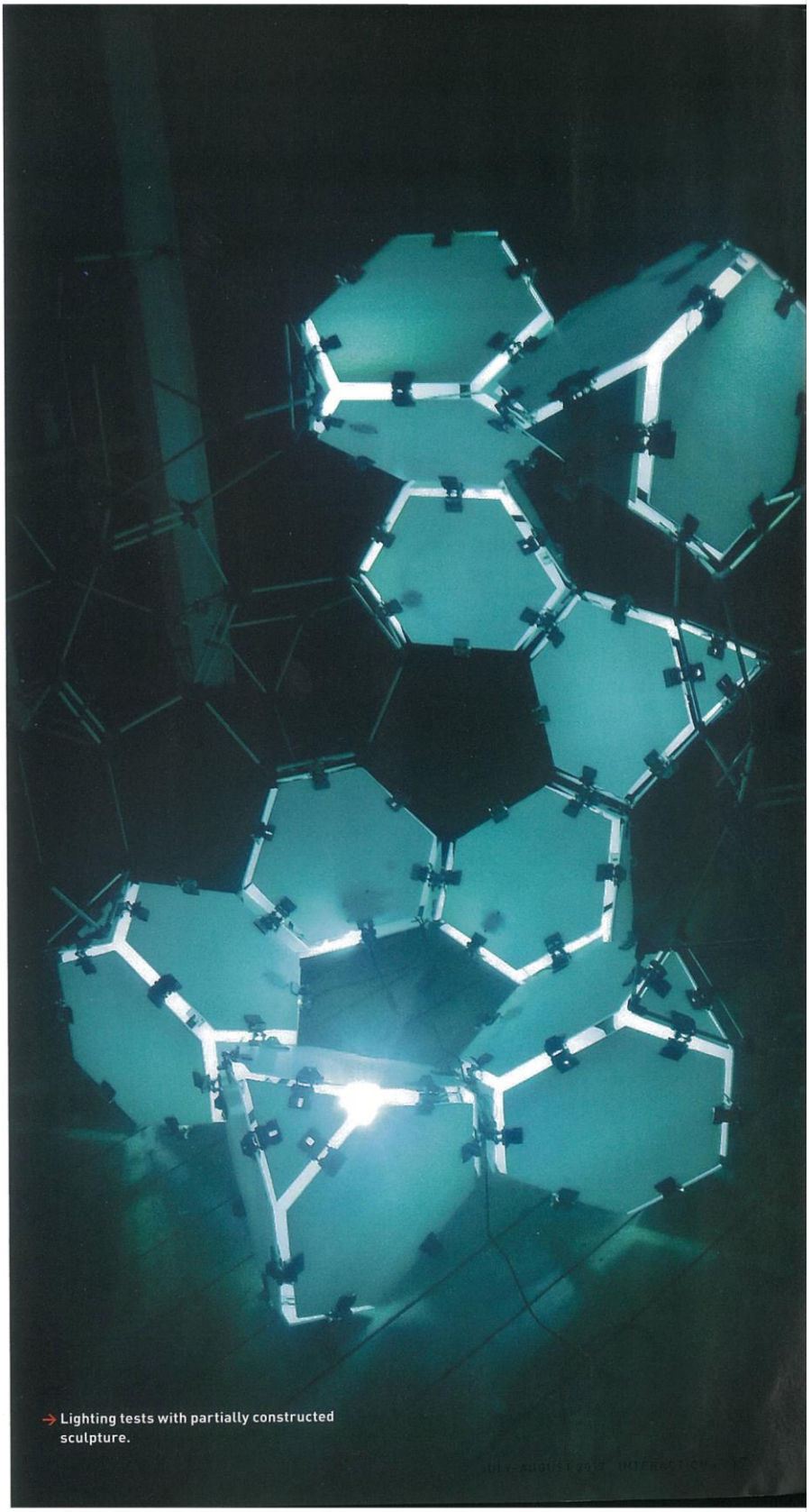
Robert Cameron,
The University of Western Australia
→ thedeadpixelproject@outlook.com

http://www.thedeadpixelproject.com/subpages/research/cringeMACHINE.html

DOI: 10.1145/3095803



→ **Detail of completed sculpture responding to sound.**



→ **Lighting tests with partially constructed sculpture.**

C. Project scripts

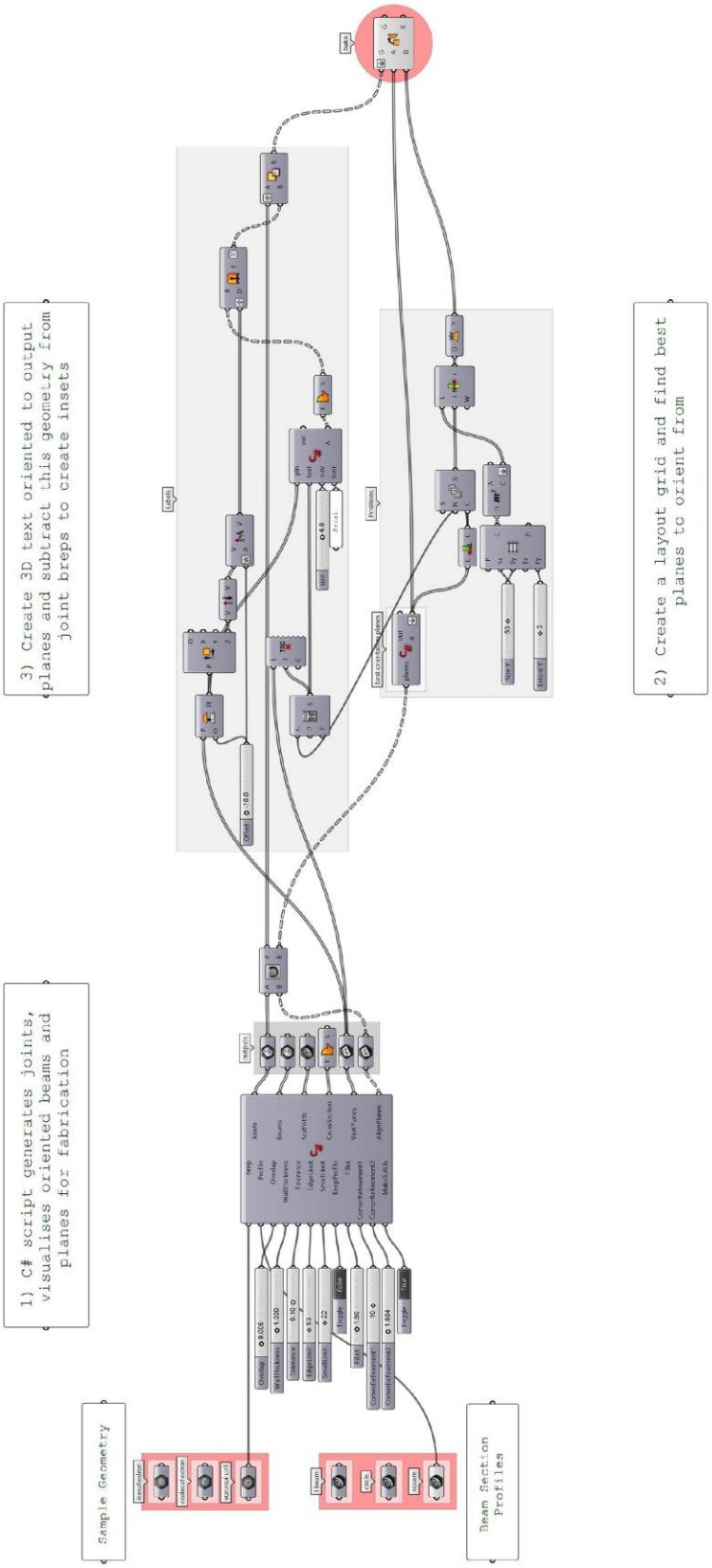
C# grasshopper component for Cellular Form project generation:

```
                                Cellular Aggregator.txt
  private void RunScript(Brep obj, int steps, List<Brep> walls, int seed, ref object
A, ref object B, ref object C)
  {

    //declare first Brep:
    Brep startB = obj;

    Lst = new List<Brep>();
    Lst.Add(startB);
    IntLst = new List<Brep>();
    bounds = new List<Brep>(walls);


    //declare scaled Brep (for intersection testing)
    Brep copy = new Brep();
    copy = startB.DuplicateBrep();
    VolumeMassProperties vmp = VolumeMassProperties.Compute(copy);
    Point3d pt = vmp.Centroid;
    Transform xformSc = new Transform();
    xformSc = Transform.Scale(pt, 0.9);
    copy.Transform(xformSc);
    IntLst.Add(copy);

    //perform main recurring method
    loop(seed, steps);


    //output
    A = IntLst;
    B = Lst;


  }

  // <Custom additional code>
  List<Brep> bounds;
  List<Brep> IntLst;
  List<Brep> Lst;
  Random random = new Random(DateTime.Now.Ticks.GetHashCode());
  int safety = 0;//safety exit counter

  static int R(int r, int s, int e)
  {
    Random rnd = new Random(r);
    return rnd.Next(s, e);
  }


  // Main Method
  public void loop(int rand, int count)
  {
    //Print("main loop: " + count.ToString());//temp

    if(count > 0)
    {
      count -= 1;
      Brep temp2;
```

```
      Brep temp1;
      addB(rand, out temp1, out temp2);

      //if(intB(temp2)){Print("true");}else{Print("false");}

      if(intB(temp2) || intW(temp2))
      {count += 1;
        safety = +1;
      }else{
        IntLst.Add(temp2);
        Lst.Add(temp1);
        safety = 0;
      }

      if(safety == temp2.Surfaces.Count()){return;}//sagety program extit for trapped
breps

      loop(rand + 1, count);
    }
  }


  // random walk Brep addition
  public void addB(int rand, out Brep copy, out Brep copy2)
  {
    Brep temp = Lst[Lst.Count() - 1];
    int dice = R(rand, 0, temp.Surfaces.Count());

    Plane frame = new Plane();
    temp.Surfaces[dice].FrameAt(0.5, 0.5, out frame);

    Transform xform = new Transform();
    xform = Transform.Mirror(frame);

    copy = temp.DuplicateBrep();

    copy.Transform(xform);
    copy2 = copy.DuplicateBrep();
    VolumeMassProperties vmp = VolumeMassProperties.Compute(copy2);
    Point3d pt = vmp.Centroid;
    Transform xformSc = new Transform();
    xformSc = Transform.Scale(pt, 0.9);
    copy2.Transform(xformSc);
  }


  // Brep intersection test
  public bool intB(Brep test)
  {

    Curve[] crv;
    Point3d[] pts;

    bool fin = false;


    for (int i = 0; i < IntLst.Count() - 1; i++)
```

```
    {
      Brep temp = new Brep();
      temp = IntLst[i];
      Rhino.Geometry.Intersect.Intersection.BrepBrep(temp, test, 0.01, out crv, out
pts);
      if(crv.Length > 0 | pts.Length > 0)
      {
        fin = true;
        Print("int happened");
      }
    }

    return fin;
  }


  // Brep intersection wall
  public bool intW(Brep test)
  {

    Curve[] crv;
    Point3d[] pts;
    //
    bool fin = false;
    //
    //
    foreach(Brep b in bounds)
    {
      //      Brep temp = new Brep();
      //      temp = IntLst[i];
      Rhino.Geometry.Intersect.Intersection.BrepBrep(b, test, 0.01, out crv, out pts);
      if(crv.Length > 0 | pts.Length > 0)
      {
        fin = true;
        Print("wall intersection");
      }
    }

    return fin;
  }
  // </Custom additional code>
}
```

Grasshopper Frame Script for Mediated Form project tectonics and layout (improved):



C# Frame Script for Mediated Form project tectonics:

```
                                    Frame Script.txt
  private void RunScript(Brep brep, Curve Profile, double Overlap, double
WallThickness, double Tolerance,
        double EdgeLimit, double SmallLimit, bool KeepProfile, double Fillet, int
CornerRefinement1,
        double CornerRefinement2, bool MakeSolids, ref object Joints, ref object
Beams, ref object Scaffolds,
        ref object CrossSection, ref object VertPlanes, ref object AlignPlanes)
  {
    edges = new List<Curve>();
    verts = new List<Point3d>();
    edges = brep.DuplicateEdgeCurves().ToList();
    verts = brep.DuplicateVertices().ToList();

    nodes = new List<Node>();
    beams = new List<Beam>();

    extrusions = new List<Brep>();  //beam visualisation
    PC = new List<Curve>();         //profile curve visualisation
    planes = new List<Plane>();     //labelling planes
    breps = new List<Brep>();       //joints
    scaffolds = new List<Curve>();  //joint visualisation
    align = new DataTree<Plane>();  //alignment planes

    //key methods::
    buildBeams(brep, Profile, Overlap, WallThickness, Tolerance, EdgeLimit,
SmallLimit, Fillet, KeepProfile);
    structJoints(brep, EdgeLimit);
    buildJoints(Overlap, CornerRefinement1, CornerRefinement2);
    brepJoints(MakeSolids, EdgeLimit);

    //Outputs::
    Joints = breps;
    Scaffolds = scaffolds;
    Beams = extrusions;
    CrossSection = PC;
    VertPlanes = planes;
    AlignPlanes = align;
  }

  // <Custom additional code>
  List<Curve> edges;
  List<Point3d> verts;
  List<Node> nodes;

  List<Beam> beams;       //beam visualisation
  List<Brep> extrusions;  //beam visualisation
  List<Curve> PC;         //profile curve visualisation
  List<Plane> planes;     //planes for text reference inprint
  List<Curve>scaffolds;   //helps to visualise joints before constructing breps
  List<Brep> breps;       //joint breps
  DataTree<Plane>align;       //debug planes

  double tol = Rhino.RhinoDoc.ActiveDoc.ModelAbsoluteTolerance;
  List<int> usedVerts = new List<int>();

  struct Node
  {
                                    Page 1
```

```
    public List<Beam> beams;
    public List<int> vertex;        //all verticies of joint
    public List<Curve> curvesA;     //corner nurbs top
    public List<Curve> curvesB;
    public List<Curve> curvesC;
    public List<Curve> curvesD;     //corner nurbs bottom
    public List<Curve> LinesTop;    //cap lines top
    public List<Curve> LinesBottom;
    public List<Curve> LinesLeft;
    public List<Curve> LinesRight;  //cap lines bottom
    public List<Plane> capPlanes;   //projection pln to correct for patch inaccuracies
    public List<Curve> insets;      //interior profiles of caps
  }

  struct Beam
  {
    public sbyte type;
    //types:
    //0: smaller than limit and (profile)width*3,
    //1: smaller than limit,
    //2: normal
    public int index;
    public Point3d start;
    public Point3d end;
    public Vector3d startVec;
    public Vector3d endVec;
    public Vector3d normal;
    public int startVert;
    public int endVert;

    //for solid construction::
    public List<Point3d> StartPoints; //points at start
    public List<Point3d> EndPoints; //points at end

    public Curve StartInset;
    public Curve EndInset;
    public Plane StartPlane;
    public Plane EndPlane;
  }

  //construct beams
  void buildBeams(Brep brep, Curve Profile, double Overlap, double WallThickness,
    double Tolerance, double EdgeLimit, double SmallLimit, double Fillet, bool
KeepProfile)
  {
    //profile measurment:
    BoundingBox bb = Profile.GetBoundingBox(Plane.WorldXY);
    Point3d[] corners = bb.GetCorners();
    double width = corners[0].DistanceTo(corners[3]);

    #region extra profiles:
    List<Curve> tempLst = new List<Curve>();
    List<Curve> outsideSection = new List<Curve>();   //outside section curve
    List<Point3d> Cpt = new List<Point3d>();
    List<Point3d> ends = new List<Point3d>();         //joint construction points

    Cpt.Add(corners[0]);Cpt.Add(corners[1]);Cpt.Add(corners[2]);Cpt.Add(corners[3]);
```

```
 Cpt.Add(corners[0]);
    Curve inHole;   //inHole (offset)
    if(!KeepProfile){inHole = Curve.CreateInterpolatedCurve(Cpt, 1);}else{inHole =
Profile;}
    if(Tolerance != 0)inHole = inHole.Offset(Plane.WorldXY, Tolerance, tol,
CurveOffsetCornerStyle.Sharp)[0];

    corners[0].X -= WallThickness + Tolerance;corners[0].Y -= WallThickness +
Tolerance;
    corners[1].X += WallThickness + Tolerance;corners[1].Y -= WallThickness +
Tolerance;
    corners[2].X += WallThickness + Tolerance;corners[2].Y += WallThickness +
Tolerance;
    corners[3].X -= WallThickness + Tolerance;corners[3].Y += WallThickness +
Tolerance;
    Point3d ptA = corners[0];Point3d ptB = corners[0];
    Point3d ptC = corners[1];Point3d ptD = corners[1];
    Point3d ptE = corners[2];Point3d ptF = corners[2];
    Point3d ptG = corners[3];Point3d ptH = corners[3];
    ptA.Y += Fillet;ptB.X += Fillet;
    ptC.X -= Fillet;ptD.Y += Fillet;
    ptE.Y -= Fillet;ptF.X -= Fillet;
    ptG.X += Fillet;ptH.Y -= Fillet;
    ends.Add(ptG);ends.Add(ptH);ends.Add(ptA);ends.Add(ptB);
    ends.Add(ptC);ends.Add(ptD);ends.Add(ptE);ends.Add(ptF);
    tempLst.Add(new Line(ptH, ptA).ToNurbsCurve());
    tempLst.Add(new Line(ptB, ptC).ToNurbsCurve());
    tempLst.Add(new Line(ptD, ptE).ToNurbsCurve());
    tempLst.Add(new Line(ptF, ptG).ToNurbsCurve());
    tempLst.Add(new Arc(ptA, -Vector3d.YAxis, ptB).ToNurbsCurve());
    tempLst.Add(new Arc(ptC, Vector3d.XAxis, ptD).ToNurbsCurve());
    tempLst.Add(new Arc(ptE, Vector3d.YAxis, ptF).ToNurbsCurve());
    tempLst.Add(new Arc(ptG, -Vector3d.XAxis, ptH).ToNurbsCurve());

    PC.Add(Curve.JoinCurves(tempLst)[0]);
    PC.Add(inHole);

    #endregion

    //find offsets:
    for (int e = 0; e < edges.Count(); e++)
    {
      Beam beam = new Beam();

      if(edges[e].GetLength() <= EdgeLimit && edges[e].GetLength() <=
SmallLimit)//width * 3.7)
      {beam.type = 0;}
      else if(edges[e].GetLength() <= EdgeLimit)
      {beam.type = 1;}
      else
      {beam.type = 2;}

      beam.normal = Normal(edges[e].PointAtNormalizedLength(0.5), brep);
      beam.index = e;

      for (int v = 0; v < verts.Count(); v++)
      {
```

```
        if(verts[v].DistanceTo(edges[e].PointAtStart) <= tol)
        {
          if(beam.type == 2)
          {
            beam.start = offset(e, v, true, Tolerance, WallThickness, width) +
edges[e].PointAtStart;
            beam.startVec = offset(e, v, true, Tolerance, WallThickness, width);
          }else
          {
            Vector3d midMove = edges[e].PointAtEnd - edges[e].PointAtStart;
            beam.startVec = midMove * (midMove.Length * 0.4);
          }
          beam.startVert = v;
        }

        if(verts[v].DistanceTo(edges[e].PointAtEnd) <= tol)
        {
          if(beam.type == 2)
          {
            beam.end = offset(e, v, false, Tolerance, WallThickness, width) +
edges[e].PointAtEnd;
            beam.endVec = offset(e, v, false, Tolerance, WallThickness, width);
          }else
          {
            Vector3d midMove = edges[e].PointAtStart - edges[e].PointAtEnd;
            beam.endVec = midMove * (midMove.Length * 0.4);
          }
          beam.endVert = v;
        }
      }

      #region fill type 2
      if(beam.type == 2){
        //Construct Breps::
        Plane plnE = new Plane(beam.end, beam.start - beam.end);
        plnE.Rotate(Vector3d.VectorAngle(plnE.YAxis, beam.normal, plnE), plnE.ZAxis);
        Curve tempE = Profile.DuplicateCurve();
        Transform insideE = Transform.ChangeBasis(plnE, Plane.WorldXY);
        tempE.Transform(insideE);
        Brep extrusion = Extrusion.Create(tempE, (beam.start - beam.end).Length,
true).ToBrep();
        extrusions.Add(extrusion);

        //END POINTS::
        //inside profiles
        Vector3d tVec = beam.end - beam.start;
        tVec.Unitize();
        tVec *= Tolerance;
        Plane plnA = new Plane(beam.end + tVec, beam.start - beam.end);
        plnA.Rotate(Vector3d.VectorAngle(plnA.YAxis, beam.normal, plnA), plnA.ZAxis);
        Curve tempA = inHole.DuplicateCurve();
        Transform inside = Transform.ChangeBasis(plnA, Plane.WorldXY);
        tempA.Transform(inside);

        //end planes
        Plane plnC = plnA;
        Vector3d vecA = beam.endVec;
```

```
vecA.Unitize();
vecA *= Overlap + Tolerance - 1;
plnC.Translate(vecA);
Transform Eoutside = Transform.ChangeBasis(plnC, Plane.WorldXY);

vecA.Unitize();
plnC.Translate(vecA);

//Point Orientation::
List<Point3d> EndPt = new List<Point3d>();
foreach(Point3d pt in ends)
{
  Point3d temp = new Point3d();
  temp = pt;
  temp.Transform(Eoutside);
  EndPt.Add(temp);
}

beam.EndInset = tempA;
beam.EndPlane = plnC;
beam.EndPoints = EndPt;

//START POINTS::
//inside profiles
Vector3d tSVec = beam.start - beam.end;
tSVec.Unitize();
tSVec *= Tolerance;
Plane plnB = new Plane(beam.start + tSVec, beam.end - beam.start);
plnB.Rotate(Vector3d.VectorAngle(plnB.YAxis, beam.normal, plnB), plnB.ZAxis);
Curve tempB = inHole.DuplicateCurve();
Transform outside = Transform.ChangeBasis(plnB, Plane.WorldXY);
tempB.Transform(outside);

//start planes
Plane plnD = plnB;
Vector3d vecB = beam.startVec;
vecB.Unitize();
vecB *= Overlap + Tolerance - 1;
plnD.Translate(vecB);
Transform Soutside = Transform.ChangeBasis(plnD, Plane.WorldXY);

vecB.Unitize();
plnD.Translate(vecB);

//Point Orientation::
List<Point3d> StrPt = new List<Point3d>();
foreach(Point3d pt in ends)
{
  Point3d temp = new Point3d();
  temp = pt;
  temp.Transform(Soutside);
  StrPt.Add(temp);
}

beam.StartInset = tempB;
beam.StartPlane = plnD;
beam.StartPoints = StrPt;
```

```
}
#endregion

#region fill type 1
if(beam.type == 1)
{
  Vector3d vecA = edges[e].PointAtStart - edges[e].PointAtEnd;
  Vector3d vecB = edges[e].PointAtEnd - edges[e].PointAtStart;
  vecA.Unitize();vecB.Unitize();

  //END::
  Plane plnA = new Plane(edges[e].PointAtNormalizedLength(0.5) + vecB, vecA);
  plnA.Rotate(Vector3d.VectorAngle(plnA.YAxis, beam.normal, plnA), plnA.ZAxis);
  Transform Eoutside = Transform.ChangeBasis(plnA, Plane.WorldXY);
  plnA.Translate(vecA);
  beam.EndPlane = plnA; //populate

  Vector3d vecC = new Vector3d();
  Plane plnE = new Plane();
  vecC = vecB;
  plnE = plnA;
  vecC *= edges[e].GetLength() * 0.3;
  plnE.Translate(vecC);
  Transform trA = Transform.ChangeBasis(plnE, Plane.WorldXY);
  Curve cA = inHole.DuplicateCurve();
  cA.Transform(trA);
  beam.EndInset = cA;   //populate

  //Point Orientation::
  List<Point3d> EndPt = new List<Point3d>();
  foreach(Point3d pt in ends)
  {
    Point3d temp = new Point3d();
    temp = pt;
    temp.Transform(Eoutside);
    EndPt.Add(temp);
  }

  //START::
  Plane plnB = new Plane(edges[e].PointAtNormalizedLength(0.5) + vecA, vecB);
  plnB.Rotate(Vector3d.VectorAngle(plnB.YAxis, beam.normal, plnB), plnB.ZAxis);
  Transform Einside = Transform.ChangeBasis(plnB, Plane.WorldXY);
  plnB.Translate(vecB);
  beam.StartPlane = plnB; //populate

  Vector3d vecD = new Vector3d();
  Plane plnD = new Plane();
  vecD = vecA;
  plnD = plnB;
  vecD *= edges[e].GetLength() * 0.3;
  plnD.Translate(vecD);
  Transform trB = Transform.ChangeBasis(plnD, Plane.WorldXY);
  Curve cB = inHole.DuplicateCurve();
  cB.Transform(trB);
  beam.StartInset = cB;   //populate

  //Point Orientation::
```

```
      List<Point3d> StrPt = new List<Point3d>();
      foreach(Point3d pt in ends)
      {
        Point3d temp = new Point3d();
        temp = pt;
        temp.Transform(Einside);
        StrPt.Add(temp);
      }

      beam.StartPlane = plnB;
      beam.EndPoints = EndPt;
      beam.StartPoints = StrPt;
    }
    #endregion


    //assign beam
    beams.Add(beam);
  }
}

//construct joint data structure
void structJoints(Brep brep, double EdgeLimit)
{
  usedVerts.Clear();
  for (int v = 0; v < verts.Count(); v++)
  {
    Node nd = new Node();
    List<Beam> adjacent = new List<Beam>();
    List<Beam> adjacent2 = new List<Beam>();
    bool Prong = false;
    int index = 0;  //potential vert 2 index
    int prongBreak = 0; //index in list for list 2 insertion
    Plane plnB = new Plane(); //potential plane for vertex 2
    Vector3d orient = new Vector3d(); //to orient second plane back

    foreach (int i in usedVerts)
    {
      if(i == v)goto Finish;
    }

    for (int e = 0; e < beams.Count(); e++) //find joining beams
    {
      sortOut = new List<Beam>();
      if(beams[e].startVert == v || beams[e].endVert == v)
      {
        if(!Prong && beams[e].type == 0)//setup
        {
          Prong = true; //happens once/vert loop
          prongBreak = e;

          if(beams[e].startVert == v)
          {
            index = beams[e].endVert;
            orient = beams[e].endVec;
          }
          if(beams[e].endVert == v)
```

```
          {
            index = beams[e].startVert;
            orient = beams[e].startVec;
          }
          usedVerts.Add(index);//prevent from going over the same verts

          plnB = new Plane(verts[index], Normal(verts[index], brep));
          plnB.Rotate(Vector3d.VectorAngle(plnB.YAxis, orient, plnB), plnB.ZAxis);

          for (int f = 0; f < beams.Count(); f++)
          {
            if(beams[f].startVert == index || beams[f].endVert == index)
            {
              if(e != f){adjacent2.Add(beams[f]);}
            }
          }
          continue;
        }
        adjacent.Add(beams[e]);
      }
    }

    Plane plnA = new Plane(verts[v], Normal(verts[v], brep));
    nd.vertex = new List<int>();
    nd.beams = new List<Beam>();
    nd.curvesA = new List<Curve>();
    nd.curvesB = new List<Curve>();
    nd.curvesC = new List<Curve>();
    nd.curvesD = new List<Curve>();
    nd.capPlanes = new List<Plane>();
    nd.insets = new List<Curve>();
    nd.LinesTop = new List<Curve>();nd.LinesBottom = new List<Curve>();
    nd.LinesLeft = new List<Curve>();nd.LinesRight = new List<Curve>();

    if(Prong)
    {
      orient.Reverse();
      plnA.Rotate(Vector3d.VectorAngle(plnA.YAxis, orient, plnA), plnA.ZAxis);
      sort(plnA, v, adjacent);
      sortOut.Add(beams[prongBreak]);
      sort(plnB, index, adjacent2);
      sortOut.Add(beams[prongBreak]);
      nd.vertex.Add(v);
      nd.vertex.Add(index);
    }else{
      sort(plnA, v, adjacent);
      nd.vertex.Add(v);
    }
    sortOut.Add(sortOut[0]);  //add tail
    planes.Add(plnA);
    nd.beams.AddRange(sortOut);
    nodes.Add(nd);

    Finish:;
  }
}
```

```
  //construct joint curves
  void buildJoints(double Overlap, int CornerRefinement1, double CornerRefinement2)
  {
    for (int n = 0; n < nodes.Count(); n++)
    {
      Line lnA = new Line();
      Line lnB = new Line();
      int cnt = nodes[n].vertex[0]; //current point
      Plane plA = new Plane();
      Plane plB = new Plane();
      Vector3d vecA = new Vector3d();
      Vector3d vecB = new Vector3d();
      GH_Path branch = new GH_Path(n); //debug

      for (int b = 0; b < nodes[n].beams.Count() - 1; b++)
      {
        Beam A = nodes[n].beams[b];         //current beam
        Beam B = nodes[n].beams[b + 1];     //1 step forward

        List<Point3d> ptsA = new List<Point3d>();
        List<Point3d> ptsB = new List<Point3d>();

        if(cnt == A.endVert)
        {
          vecA = A.endVec;
          ptsA = A.EndPoints;
          nodes[n].insets.Add(A.EndInset);
          nodes[n].capPlanes.Add(A.EndPlane);
        }
        if(cnt == A.startVert)
        {
          vecA = A.startVec;
          ptsA = A.StartPoints;
          nodes[n].insets.Add(A.StartInset);
          nodes[n].capPlanes.Add(A.StartPlane);
        }
        if(cnt == B.endVert)
        {
          vecB = B.endVec;
          ptsB = B.EndPoints;
        }
        if(cnt == B.startVert)
        {
          vecB = B.startVec;
          ptsB = B.StartPoints;
        }

        vecA.Unitize();
        vecB.Unitize();

        plA = new Plane(verts[cnt], Vector3d.Add(vecA, vecB),
Vector3d.CrossProduct(vecA, vecB));

        //insert lines into node struct::
        nodes[n].LinesTop.Add(new Line(ptsA[7], ptsA[0]).ToNurbsCurve());
        nodes[n].LinesRight.Add(new Line(ptsA[6], ptsA[5]).ToNurbsCurve());
        nodes[n].LinesLeft.Add(new Line(ptsA[1], ptsA[2]).ToNurbsCurve());
```

```
        nodes[n].LinesBottom.Add(new Line(ptsA[3], ptsA[4]).ToNurbsCurve());

        if(B.type == 0)
        {
          Beam C = nodes[n].beams[b + 2]; //2 steps forward
          Vector3d vecC = new Vector3d();

          //switch cnt
          if(cnt == nodes[n].vertex[0])
          {cnt = nodes[n].vertex[1];}else
          {cnt = nodes[n].vertex[0];}

          if(cnt == C.endVert)
          {
            vecC = C.endVec;
            ptsB = C.EndPoints;
          }
          if(cnt == C.startVert)
          {
            vecC = C.startVec;
            ptsB = C.StartPoints;
          }
          vecB.Reverse();
          vecC.Unitize();
          vecB.Unitize();

          plB = new Plane(verts[cnt], Vector3d.Add(vecC, vecB),
Vector3d.CrossProduct(vecB, vecC));

          for (int i = 0; i <= 3; i++)
          {
            Point3d ptA = new Point3d();
            Point3d ptB = new Point3d();
            ptA = ptsA[7 - i];
            ptB = ptsB[i];
            ptA = PointToPlane(ptA, plA, vecA);
            ptB = PointToPlane(ptB, plB, vecC);
            lnA = new Line(ptsA[7 - i], ptA);
            lnB = new Line(ptsB[i], ptB);

            if(i == 0)nodes[n].curvesA.Add(Connect(lnA.ToNurbsCurve(),
lnB.ToNurbsCurve(),
                CornerRefinement1, CornerRefinement2));
            if(i == 1)nodes[n].curvesB.Add(Connect(lnA.ToNurbsCurve(),
lnB.ToNurbsCurve(),
                CornerRefinement1, CornerRefinement2));
            if(i == 2)nodes[n].curvesC.Add(Connect(lnA.ToNurbsCurve(),
lnB.ToNurbsCurve(),
                CornerRefinement1, CornerRefinement2));
            if(i == 3)nodes[n].curvesD.Add(Connect(lnA.ToNurbsCurve(),
lnB.ToNurbsCurve(),
                CornerRefinement1, CornerRefinement2));
          }
          b += 1; //shift 1 steps forward

        }else{
```

```
          for (int i = 0; i <= 3; i++)
          {
            Point3d ptA = new Point3d();
            Point3d ptB = new Point3d();
            ptA = ptsA[7 - i];
            ptB = ptsB[i];
            ptA = PointToPlane(ptA, plA, vecA);
            ptB = PointToPlane(ptB, plA, vecB);
            lnA = new Line(ptsA[7 - i], ptA);
            lnB = new Line(ptsB[i], ptB);

            if(i == 0)nodes[n].curvesA.Add(Connect(lnA.ToNurbsCurve(),
lnB.ToNurbsCurve(),
                CornerRefinement1, CornerRefinement2));
            if(i == 1)nodes[n].curvesB.Add(Connect(lnA.ToNurbsCurve(),
lnB.ToNurbsCurve(),
                CornerRefinement1, CornerRefinement2));
            if(i == 2)nodes[n].curvesC.Add(Connect(lnA.ToNurbsCurve(),
lnB.ToNurbsCurve(),
                CornerRefinement1, CornerRefinement2));
            if(i == 3)nodes[n].curvesD.Add(Connect(lnA.ToNurbsCurve(),
lnB.ToNurbsCurve(),
                CornerRefinement1, CornerRefinement2));
          }
        }
      }
      //outside loop
    }
  }

  //construct joint brep
  void brepJoints(bool MakeSolids, double EdgeLimit)
  {
    for (int n = 0; n < nodes.Count(); n++)
    {
      List<Brep> parts = new List<Brep>();
      List<Curve> linesTop = new List<Curve>();
      List<Curve> linesBottom = new List<Curve>();
      List<Curve> CrvPatchTop = new List<Curve>();
      List<Curve> CrvPatchBottom = new List<Curve>();
      GH_Path branch = new GH_Path(n);

      for (int d = 0; d < nodes[n].curvesA.Count(); d++)
      {
        //Patch
        linesTop.Add(nodes[n].LinesTop[d]);
        linesBottom.Add(nodes[n].LinesBottom[d]);
        CrvPatchTop.Add(nodes[n].curvesA[d]);
        CrvPatchBottom.Add(nodes[n].curvesD[d]);
      }

      CrvPatchTop.AddRange(linesTop);
      CrvPatchBottom.AddRange(linesBottom);

      //Patch and return modified edge curves::
      if(MakeSolids)
      {
```

```
  parts.Add(Patch(ref CrvPatchTop, EdgeLimit));
  parts.Add(Patch(ref CrvPatchBottom, EdgeLimit));

  List<List<Curve>> tempA = new List<List<Curve>>();
  List<List<Curve>> tempB = new List<List<Curve>>();

  tempA = Split(CrvPatchTop, nodes[n].curvesA.Count());
  linesTop = tempA[1];
  CrvPatchTop = tempA[0];

  tempB = Split(CrvPatchBottom, nodes[n].curvesA.Count());
  linesBottom = tempB[1];
  CrvPatchBottom = tempB[0];
}

//construct arcs for fillets::
for (int c = 0; c < nodes[n].curvesB.Count(); c++)
{
  #region ruled surfaces
  //ruled surfaces
  List<Curve> crvs = new List<Curve>();
  crvs.Add(nodes[n].curvesB[c]);
  crvs.Add(nodes[n].curvesC[c]);
  Brep sweep = NurbsSurface.CreateRuledSurface(crvs[0], crvs[1]).ToBrep();
  parts.Add(sweep);
  if(!MakeSolids)breps.Add(sweep);
  #endregion

  #region arcs
  //Network
  //top
  Curve CrvB;
  Curve CrvA = Fillet(nodes[n].LinesRight[c].PointAtEnd,
    nodes[n].LinesRight[c].PointAtStart,
    linesTop[c].PointAtStart);
  //bottom
  Curve CrvC;
  Curve CrvD = Fillet(nodes[n].LinesRight[c].PointAtStart,
    nodes[n].LinesRight[c].PointAtEnd,
    linesBottom[c].PointAtEnd);

  if(c < nodes[n].curvesB.Count() - 1)
  {
    //top
    CrvB = Fillet(nodes[n].LinesLeft[c + 1].PointAtEnd,
      nodes[n].LinesLeft[c + 1].PointAtStart,
      linesTop[c + 1].PointAtEnd);
    //bottom
    CrvC = Fillet(nodes[n].LinesLeft[c + 1].PointAtStart,
      nodes[n].LinesLeft[c + 1].PointAtEnd,
      linesBottom[c + 1].PointAtStart);
  }else{
    //top tail wrap
    CrvB = Fillet(nodes[n].LinesLeft[0].PointAtEnd,
      nodes[n].LinesLeft[0].PointAtStart,
      linesTop[0].PointAtEnd);
    //bottom
```

```
          CrvC = Fillet(nodes[n].LinesLeft[0].PointAtStart,
            nodes[n].LinesLeft[0].PointAtEnd,
            linesBottom[0].PointAtStart);
        }

        //top:
        List<Curve> CrvNetworkTop = new List<Curve>();
        List<Curve> CrvNetworkBottom = new List<Curve>();
        CrvNetworkTop.Add(CrvPatchTop[c]);
        CrvNetworkTop.Add(nodes[n].curvesB[c]);
        CrvNetworkTop.Add(CrvA);
        CrvNetworkTop.Add(CrvB);

        //bottom:
        CrvNetworkBottom.Add(nodes[n].curvesC[c]);
        CrvNetworkBottom.Add(CrvPatchBottom[c]);
        CrvNetworkBottom.Add(CrvC);
        CrvNetworkBottom.Add(CrvD);

        //network curves::
        if(MakeSolids)
        {
          int error = 0;
          parts.Add(NurbsSurface.CreateNetworkSurface(CrvNetworkTop, 3, 0.0001, tol,
tol, out error).ToBrep());
          parts.Add(NurbsSurface.CreateNetworkSurface(CrvNetworkBottom, 3, 0.0001,
tol, tol, out error).ToBrep());
        }

        //output curves::
        scaffolds.AddRange(CrvNetworkBottom);
        scaffolds.AddRange(CrvNetworkTop);
        scaffolds.AddRange(CrvPatchTop);
        scaffolds.AddRange(CrvPatchBottom);
        #endregion

        #region cap profiles

        //1. outside crv
        List<Curve> IC = new List<Curve>();
        IC.Add(CrvA);
        IC.Add(CrvD);
        IC.Add(Fillet(nodes[n].LinesLeft[c].PointAtStart,
          nodes[n].LinesLeft[c].PointAtEnd,
          linesBottom[c].PointAtStart));
        IC.Add(Fillet(nodes[n].LinesLeft[c].PointAtEnd,
          nodes[n].LinesLeft[c].PointAtStart,
          linesTop[c].PointAtEnd));
        IC.Add(nodes[n].LinesLeft[c]);  IC.Add(nodes[n].LinesRight[c]);
        IC.Add(linesTop[c]); IC.Add(linesBottom[c]);
        Curve curve = Curve.JoinCurves(IC)[0];
        scaffolds.Add(curve);

        //2. inside crv
        scaffolds.Add(nodes[n].insets[c]);

        //3. planes
```

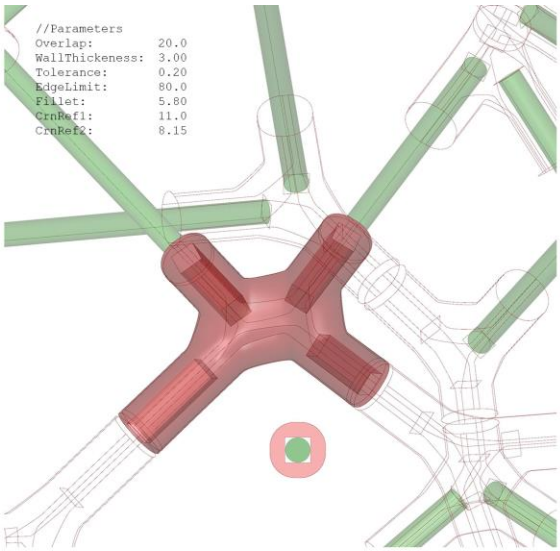
```
      align.Add(nodes[n].capPlanes[c], branch);

      //cap geometry
      if(MakeSolids)
      {
        parts.AddRange(Cap(curve, nodes[n].insets[c], nodes[n].capPlanes[c]));
      }
      #endregion
    }
    if(MakeSolids){ breps.AddRange(Brep.JoinBreps(parts, tol));}
  }
}

//Construct cap geometry
List<Brep> Cap(Curve ExteriorSection, Curve InteriorProfile, Plane AdjustmentPlane)
{
  List<Brep> lst = new List<Brep>();
  Transform trs = Transform.PlanarProjection(AdjustmentPlane);

  Curve IP = InteriorProfile.DuplicateCurve();
  IP.Transform(trs);
  Curve ES = ExteriorSection.DuplicateCurve();
  ES.Transform(trs);

  Curve[] crvA = new Curve[1]{InteriorProfile};
  Curve[] crvB = new Curve[2]{IP,ES};

  lst.AddRange(Brep.CreatePlanarBreps(crvA));
  lst.AddRange(Brep.CreatePlanarBreps(crvB));
  lst.Add(NurbsSurface.CreateRuledSurface(InteriorProfile, IP).ToBrep());
  lst.Add(NurbsSurface.CreateRuledSurface(ExteriorSection, ES).ToBrep());
  return lst;
}

//Split list
public static List<List<Curve>> Split(List<Curve> source, int size)
{
  // TODO: Prepopulate with the right capacity
  List<List<Curve>> ret = new List<List<Curve>>();
  for (int i = 0; i < source.Count; i += size)
  {
    ret.Add(source.GetRange(i, Math.Min(size, source.Count - i)));
  }
  return ret;
}

//Arcs
Curve Fillet(Point3d VectorStart, Point3d VectorEnd, Point3d ArcEnd)
{
  Arc arc = new Arc(VectorEnd, VectorEnd - VectorStart, ArcEnd);
  return arc.ToNurbsCurve();
}

//Join profiles with scaffolding curves
Curve Connect(Curve curveA, Curve curveB, int DivisionLevel, double proxi)
{
  List<Point3d> pts = new List<Point3d>();
```

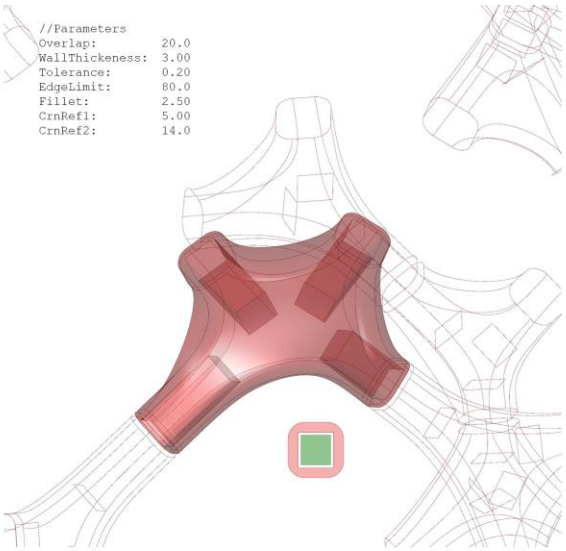
```
    List <Point3d> pts2 = new List<Point3d>();
    double total = 1.00 / DivisionLevel;

    for (int i = 0; i < DivisionLevel; i++)
    {
      double t = total * i;
      Point3d pt = curveA.PointAtNormalizedLength(t);
      Point3d pt2 = curveB.PointAtNormalizedLength(t);
      if(pt.DistanceTo(pt2) < proxi && i != 0){goto Finish;}
      pts.Add(pt);
      pts2.Add(pt2);

    }
    Finish:
      pts2.Reverse();
    pts.AddRange(pts2);
    Curve crv = Curve.CreateControlPointCurve(pts, 3);

    return crv;
  }

  //Patch method
  Brep Patch(ref List<Curve>curves, double EdgeLimit)
  {
    Brep a = Brep.CreatePatch(curves, 30, 30, tol);
    List<bool> bl = new List<bool>();
    for (int i = 0; i < 4; i++){ bl.Add(true);}
    Brep b = Brep.CreatePatch(curves, a.Surfaces[0], 30, 30, true, true, 0.1,
5/*flexibility*/, tol/*pull*/, bl.ToArray(), tol);

    Curve[] crv = b.DuplicateEdgeCurves()[0].DuplicateSegments();

    List<Curve> outLst = new List<Curve>();
    for (int i = 0; i < curves.Count(); i++)
    {
      for (int j = 0; j < crv.Count(); j++)
      {

if(curves[i].PointAtNormalizedLength(0.5).DistanceTo(crv[j].PointAtNormalizedLength(0.
5)) < 0.1)
        {
          if(curves[i].PointAtStart.DistanceTo(crv[j].PointAtStart) < 1)
          {
            outLst.Add(crv[j]);
          }else
          {
            Curve c = crv[j];
            c.Reverse();
            outLst.Add(c);
          }
        }
      }
    }
    curves = outLst;
    return b;
  }
```

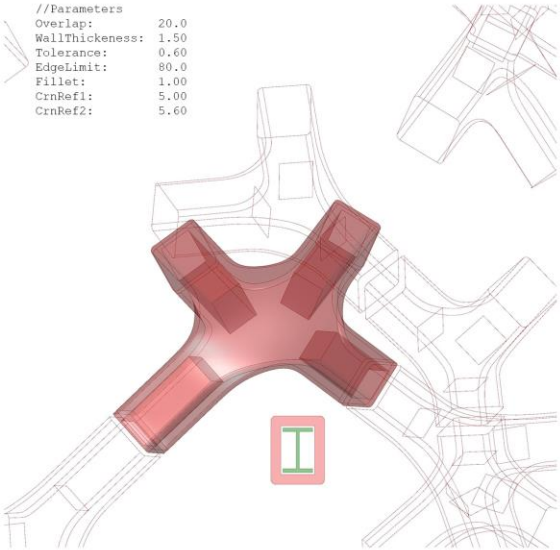
```
  //Plane|Line intersection
  Point3d PointToPlane(Point3d point, Plane plane, Vector3d direction)
  {
    Line temp = new Line(point, direction, 1);
    Point3d outPt = new Point3d();
    double par = 0;
    Rhino.Geometry.Intersect.Intersection.LinePlane(temp, plane, out par);
    outPt = temp.PointAt(par);
    return outPt;
  }

  //Vector calculations::
  Vector3d offset(int edge, int vert, bool end, double Tolerance, double
WallThickness, double width)
  {
    List<int> Nedges = new List<int>();
    Vector3d origin = new Vector3d();
    Vector3d temp = new Vector3d();
    Point3d pt = new Point3d();
    if(end)
    {
      pt = edges[edge].PointAtStart;
      origin = edges[edge].PointAtEnd - edges[edge].PointAtStart;
    }else
    {
      pt = edges[edge].PointAtEnd;
      origin = edges[edge].PointAtStart - edges[edge].PointAtEnd;
    }
    origin.Unitize();
    double ang = double.PositiveInfinity;

    for (int e = 0; e < edges.Count(); e++)
    {
      if(e == edge)continue;
      if(edges[e].PointAtStart.DistanceTo(verts[vert]) <= tol)
      {
        temp = edges[e].PointAtEnd - edges[e].PointAtStart;
        if(Vector3d.VectorAngle(origin, temp) < ang)
        {
          ang = Vector3d.VectorAngle(origin, temp);
        }
      }
      if(edges[e].PointAtEnd.DistanceTo(verts[vert]) <= tol)
      {
        temp = edges[e].PointAtStart - edges[e].PointAtEnd;
        if(Vector3d.VectorAngle(origin, temp) < ang)
        {
          ang = Vector3d.VectorAngle(origin, temp);
        }
      }
    }
    ang *= 0.5;
    double opp = width * 0.5 + WallThickness;
    ang = opp / Math.Tan(ang);
    origin *= ang;
    return origin;
  }
```

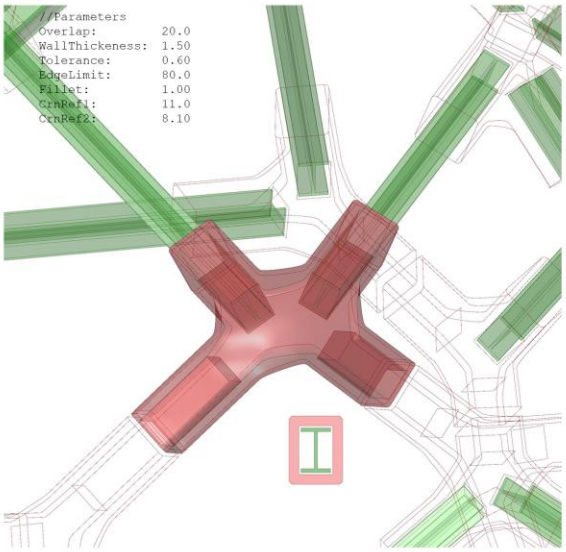
```
  //Average normal on Brep::
  Vector3d Normal(Point3d pt, Brep brep)
  {
    Vector3d norm = new Vector3d(0, 0, 0); //joint normal
    List<Surface> srf = new List<Surface>();
    srf = brep.Surfaces.ToList();
    double u = 0;
    double v = 0;

    for (int s = 0; s < srf.Count(); s++)  //go through surfaces (find normals)
    {
      srf[s].ClosestPoint(pt, out u, out v);
      if(srf[s].PointAt(u, v).DistanceTo(pt) <= tol)
      {
        norm += srf[s].NormalAt(u, v);
      }
    }
    return norm;
  }

  List<Beam>sortOut;
  //Sort lines and vectors around a plane::
  void sort(Plane plane, int vert, List<Beam> ListToSort)
  {
    List<Beam> temp = new List<Beam>();
    temp = ListToSort;

    if(temp.Count() > 0)
    {
      double outAng = double.PositiveInfinity;
      int count = 0;

      for (int i = 0; i < temp.Count(); i++)
      {
        Vector3d tempV = new Vector3d();
        if(temp[i].startVert == vert)tempV = temp[i].startVec;
        if(temp[i].endVert == vert)tempV = temp[i].endVec;

        if(Vector3d.VectorAngle(plane.YAxis, tempV, plane) < outAng)
        {
          outAng = Vector3d.VectorAngle(plane.YAxis, tempV, plane);
          count = i;
        }
      }
      Beam bm = new Beam();
      bm = temp[count];
      sortOut.Add(bm);
      temp.RemoveAt(count);
      sort(plane, vert, temp);
    }
  }

  // </Custom additional code>
}
```

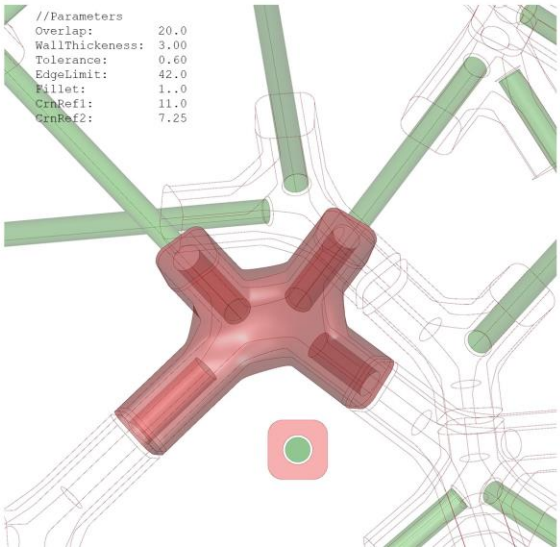
Frame Script Variations:

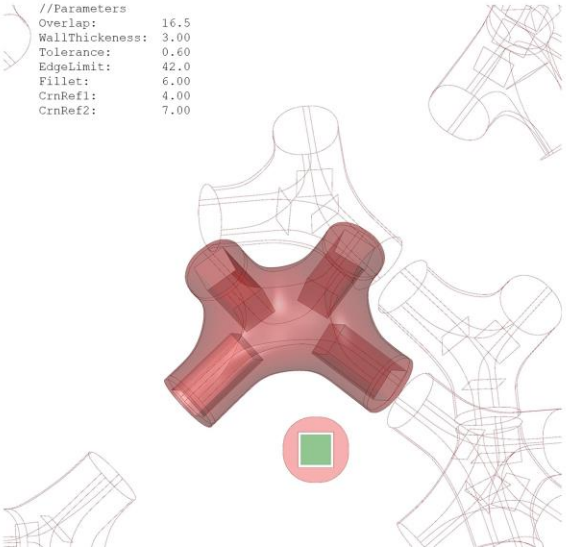




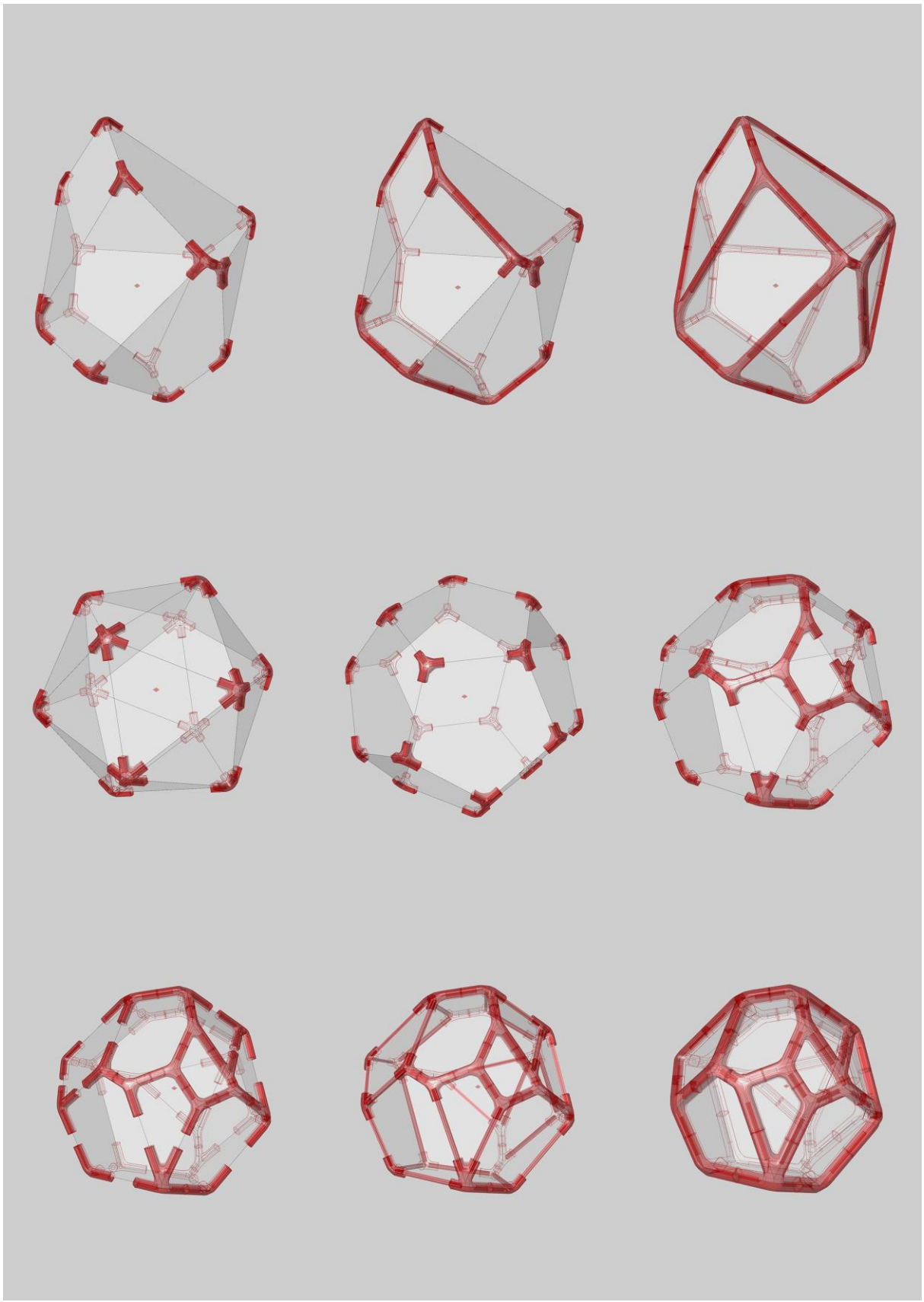




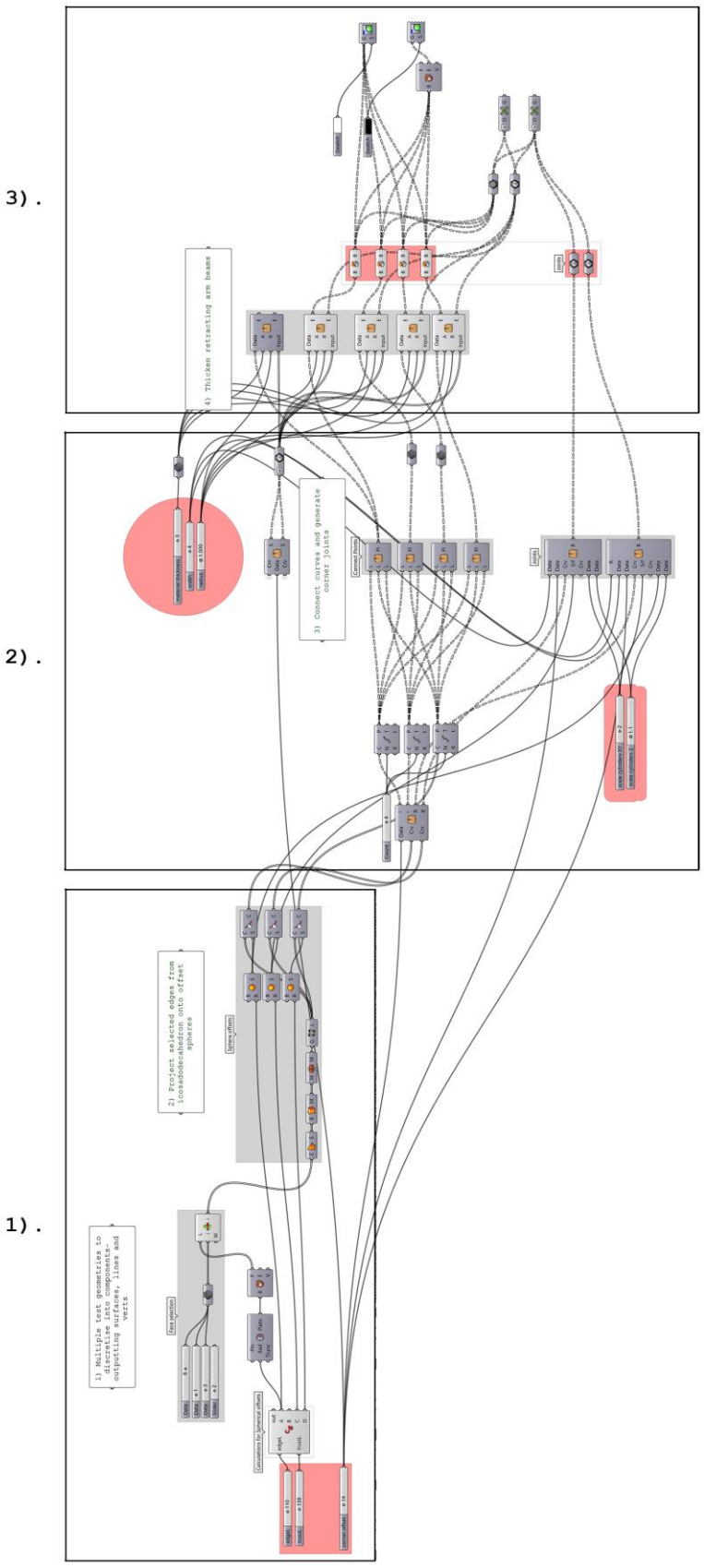




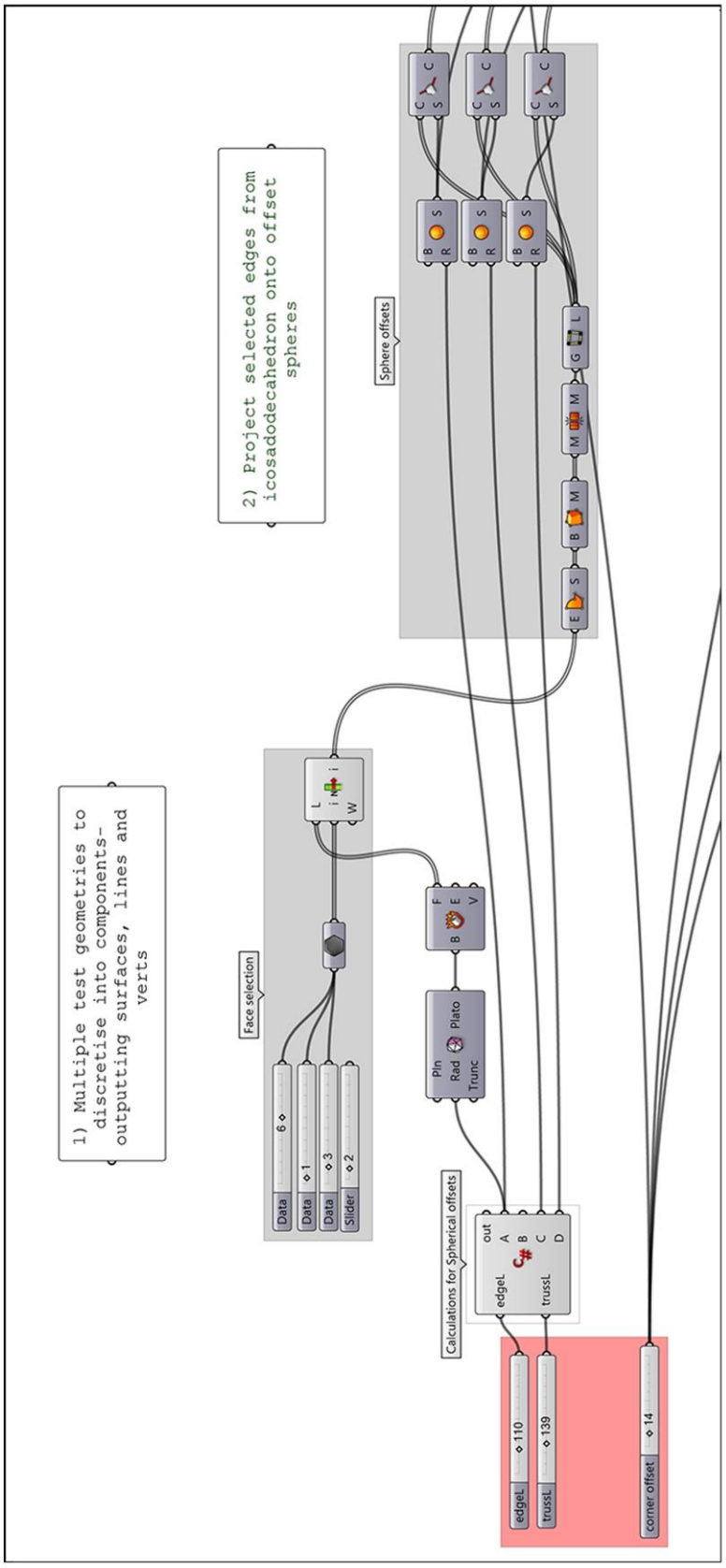
Frame script tests on a Voronoi cell:



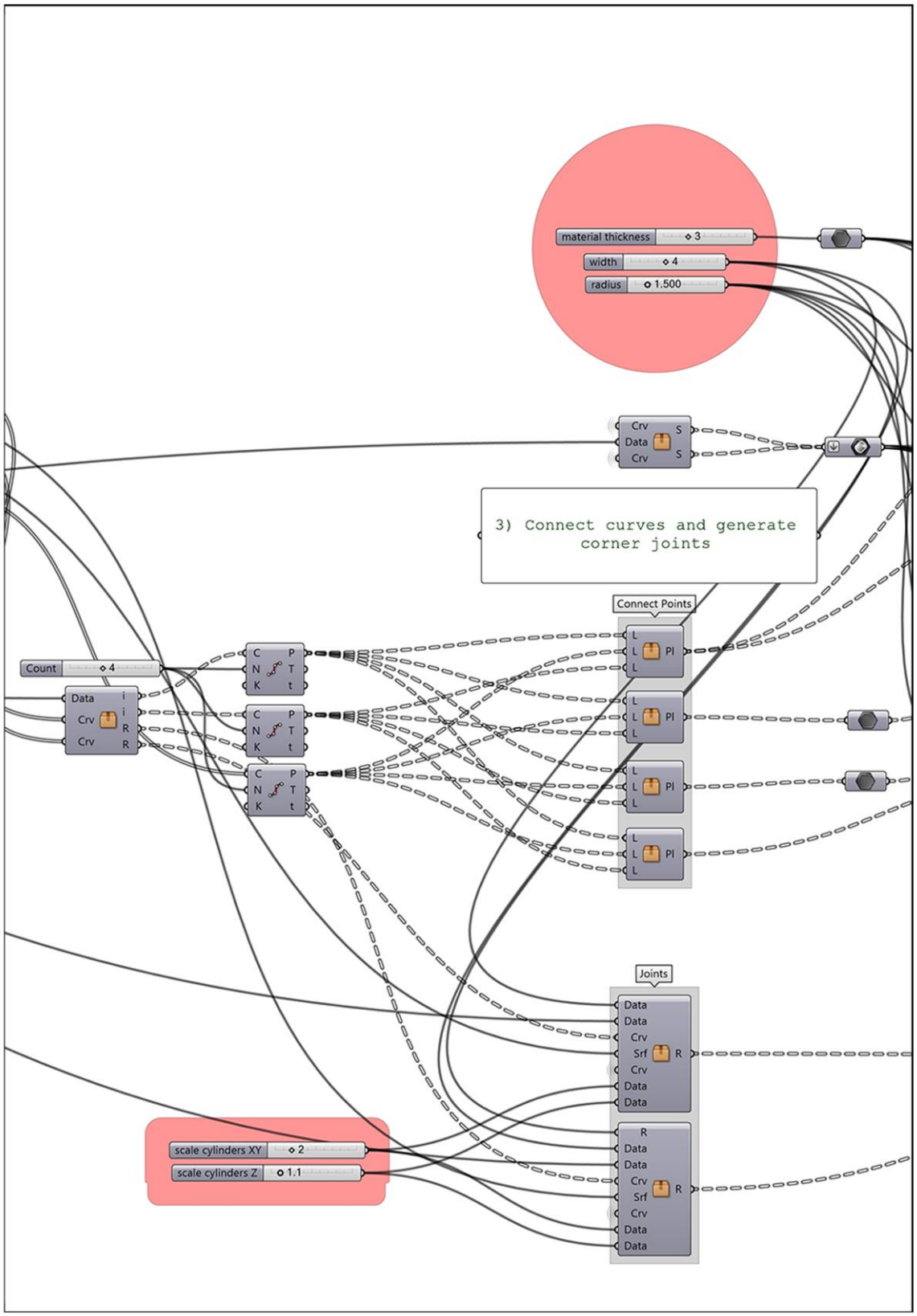
Hoberman sphere script overview:



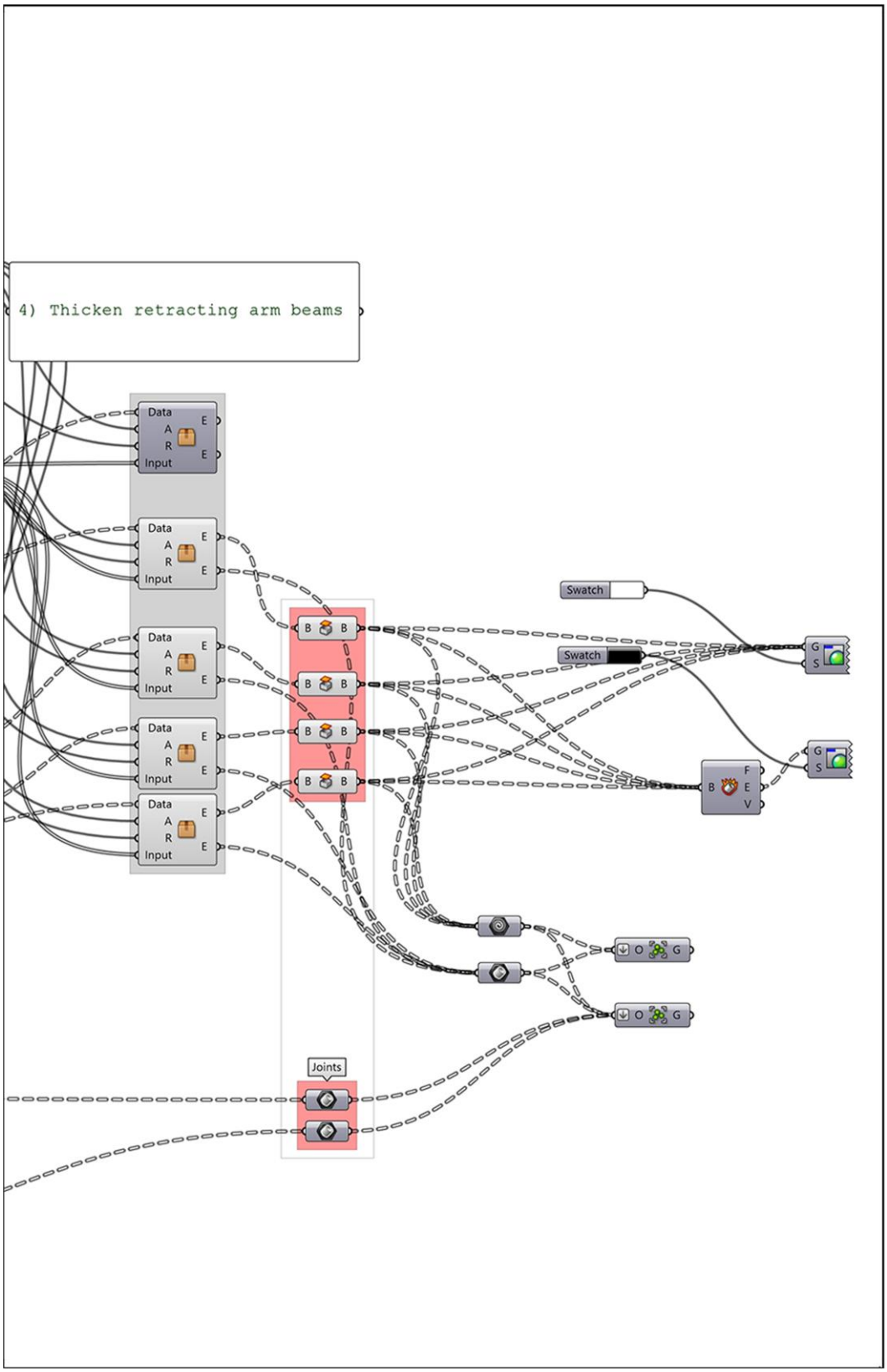
Hoberman sphere part1:



Hoberman sphere part2:



Hoberman sphere part 3:



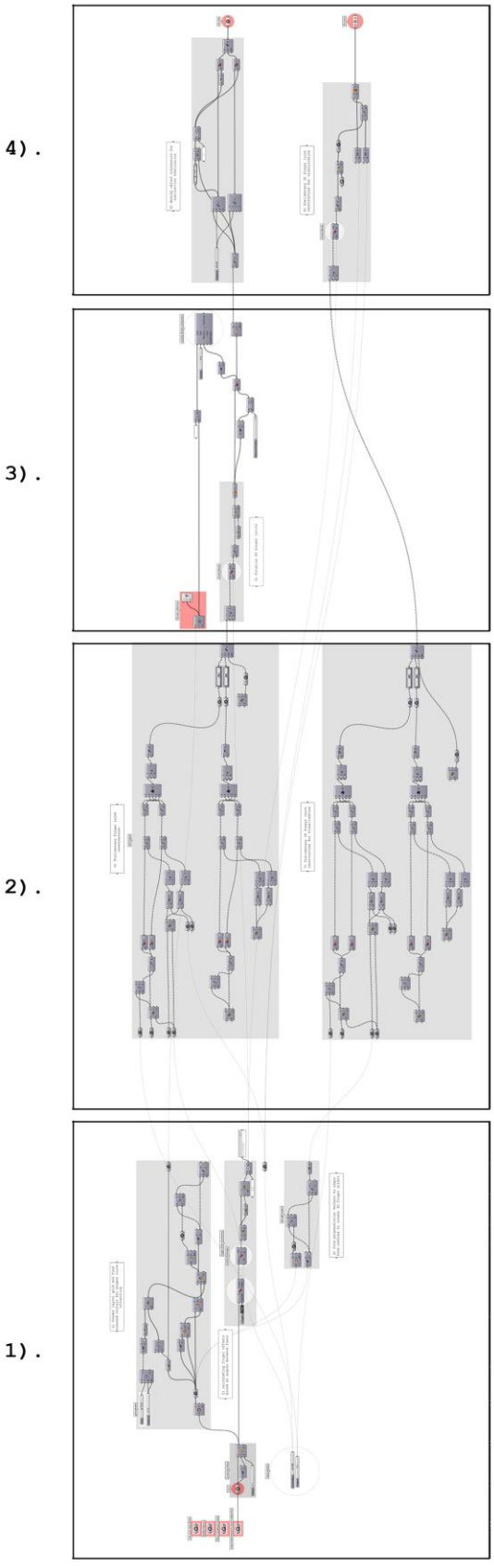
Arduino code for component design:

```
//Hbridge control

const int EN=6;
const int MC1=5;
const int MC2=4;
//const int POT=0;

const int TriggerPin = 3; //Trig pin
const int EchoPin = 10; //Echo pin
long Duration = 0;

int val=0; //for storing readings from potentiometer
int velocity=0; //for storing desired velocity

void setup()
{
  //motor
 Serial.begin(9600);
 pinMode(EN,OUTPUT);
 pinMode(MC1,OUTPUT);
 pinMode(MC2,OUTPUT);

 //echo
pinMode(TriggerPin,OUTPUT); // Trigger is an output pin
pinMode(EchoPin,INPUT); // Echo is an input pin
Serial.begin(9600); // Serial Output

 brake(); //initialise with motor disabled
}

void loop()
{
 //echo
 digitalWrite(TriggerPin, LOW);
delayMicroseconds(2);
digitalWrite(TriggerPin, HIGH); // Trigger pin to HIGH
delayMicroseconds(10); // 10us high
digitalWrite(TriggerPin, LOW); // Trigger pin to HIGH

Duration = pulseIn(EchoPin,HIGH); // Waits for the echo pin to get high
// returns the Duration in microseconds
long Distance_mm = Distance(Duration); // Use function to calculate the distance

Serial.print("Distance = "); // Output to serial
Serial.print(Distance_mm);
Serial.println(" mm");

delay(100); // Wait to do next measurement

//motor control

 val=map(Distance_mm,0,4000,255,0);
 Serial.println(val);

 if(val<255&&val>0)
 {
//go forward
```

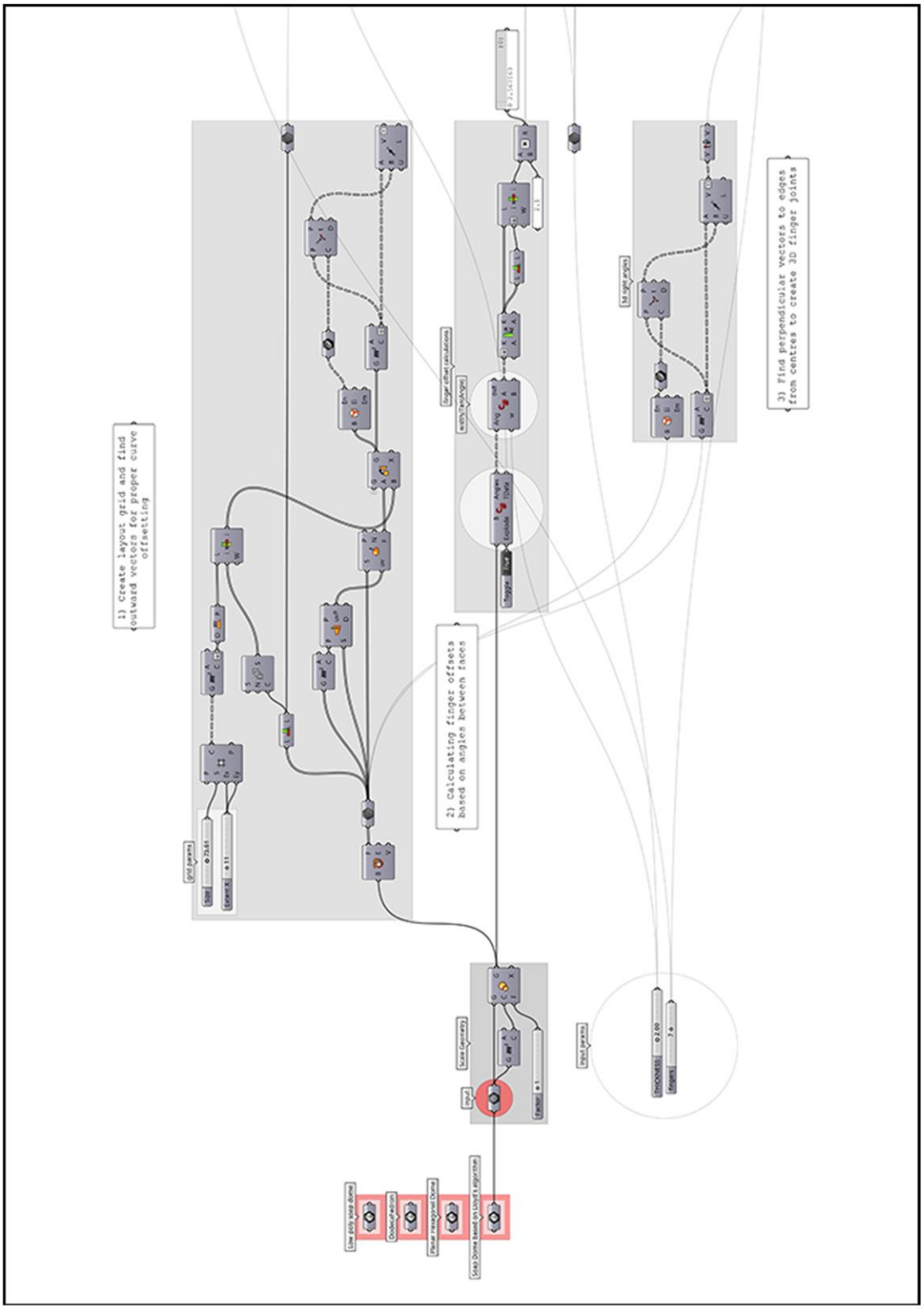
```
if(val>562)
{
  //velocity=map(val,563,1023,0,255);
  forward(val);
}

//go back
else if(val<462)
{
  //velocity=map(val,461,0,0,255);
  reverse(val);
}

//brake
else
{
 brake();
}
 }
else
{
 brake();
}
}

//classes

//motor goes forward (0-255)

void forward(int rate)
{
 digitalWrite(EN,LOW);
digitalWrite(MC1,HIGH);
digitalWrite(MC2,LOW);
analogWrite(EN,rate);
}

//motor goes backward (0-255)

void reverse(int rate)
{
 digitalWrite(EN,LOW);
digitalWrite(MC1,LOW);
digitalWrite(MC2,HIGH);
analogWrite(EN,rate);
}

//stop motor

void brake()
{
 digitalWrite(EN,LOW);
digitalWrite(MC1,LOW);
digitalWrite(MC2,LOW);
analogWrite(EN,HIGH);
}
```

```
//calculate echo distance
long Distance(long time)
{
// Calculates the Distance in mm
// ((time)*(Speed of sound))/ toward and backward of object) * 10

long DistanceCalc; // Calculation variable
DistanceCalc = ((time /2.9) / 2); // Actual calculation in mm
//DistanceCalc = time / 74 / 2; // Actual calculation in inches
return DistanceCalc; // return calculated value
}
```

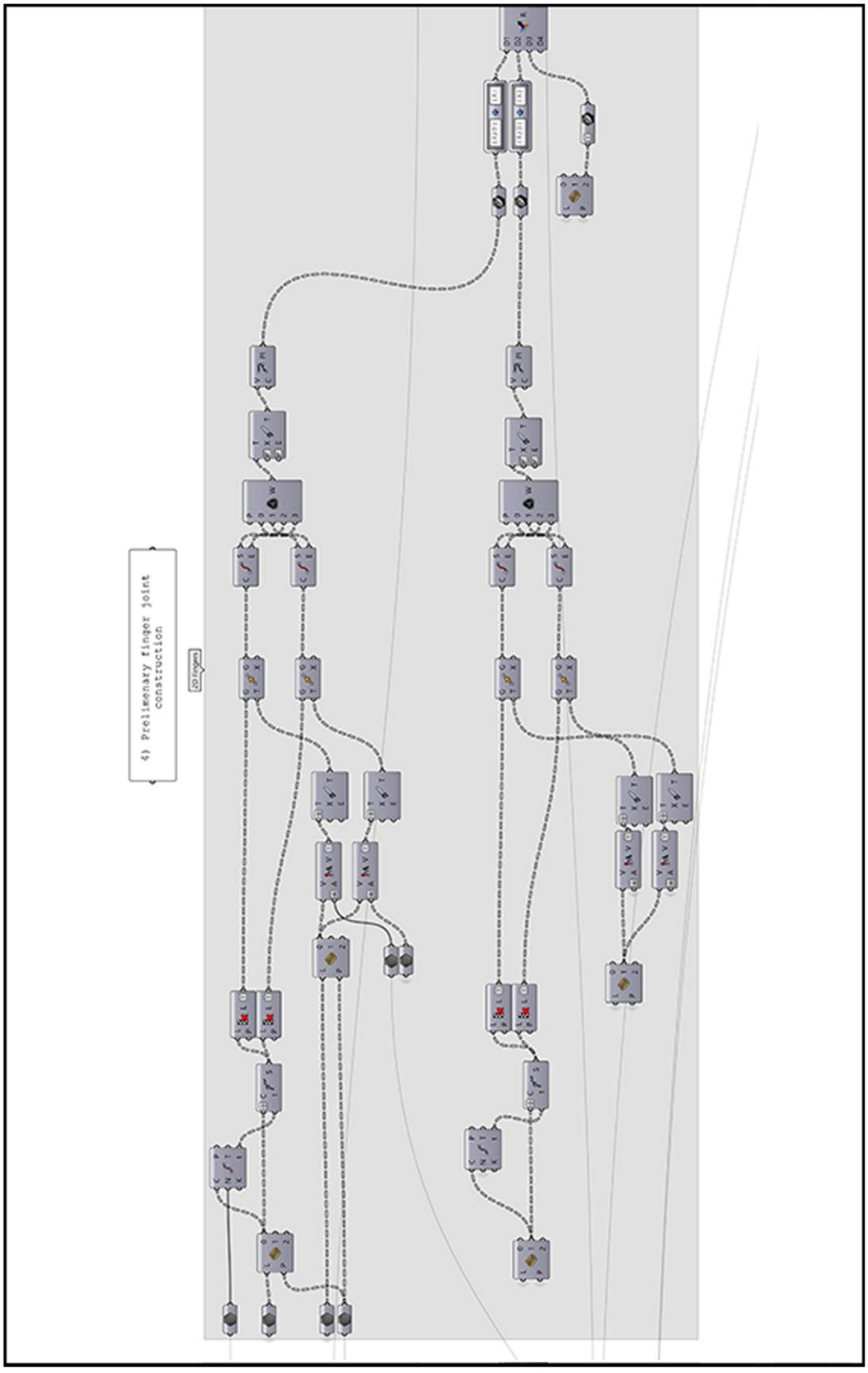
Finger joints Grasshopper script (Kinetic surface test) overview:



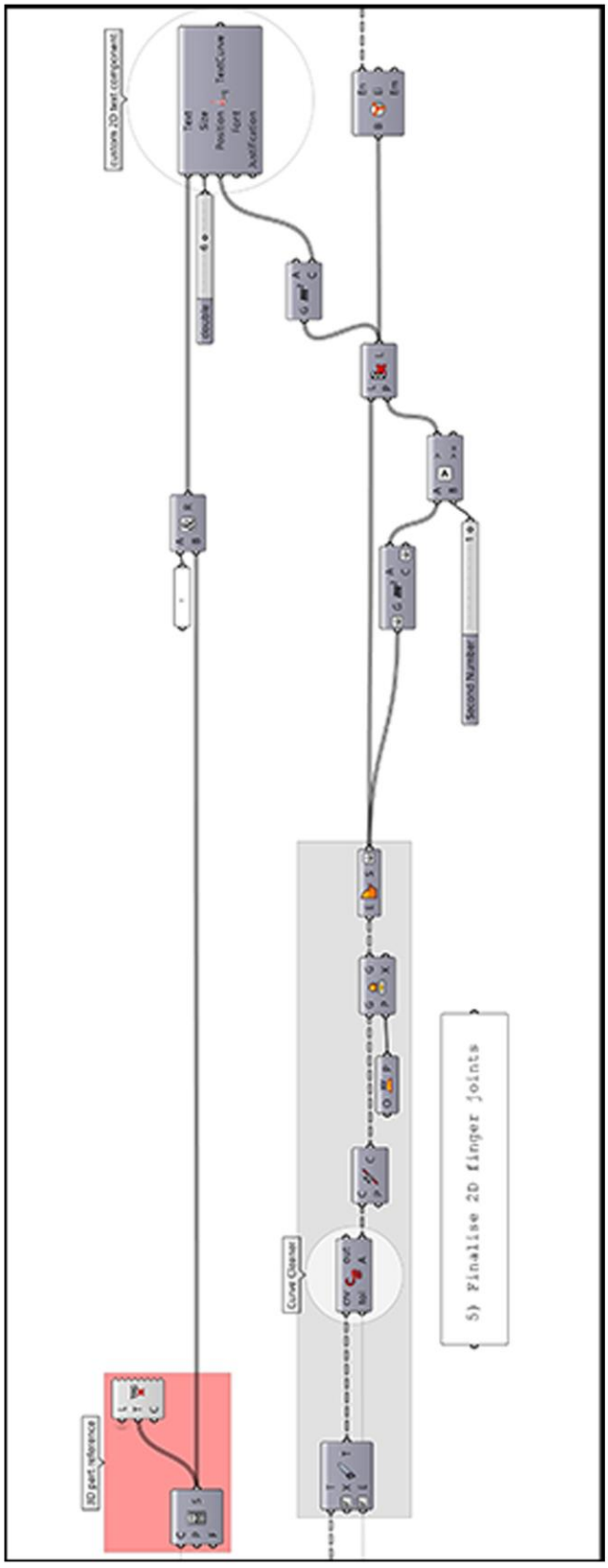
Finger joints Grasshopper script part 1:



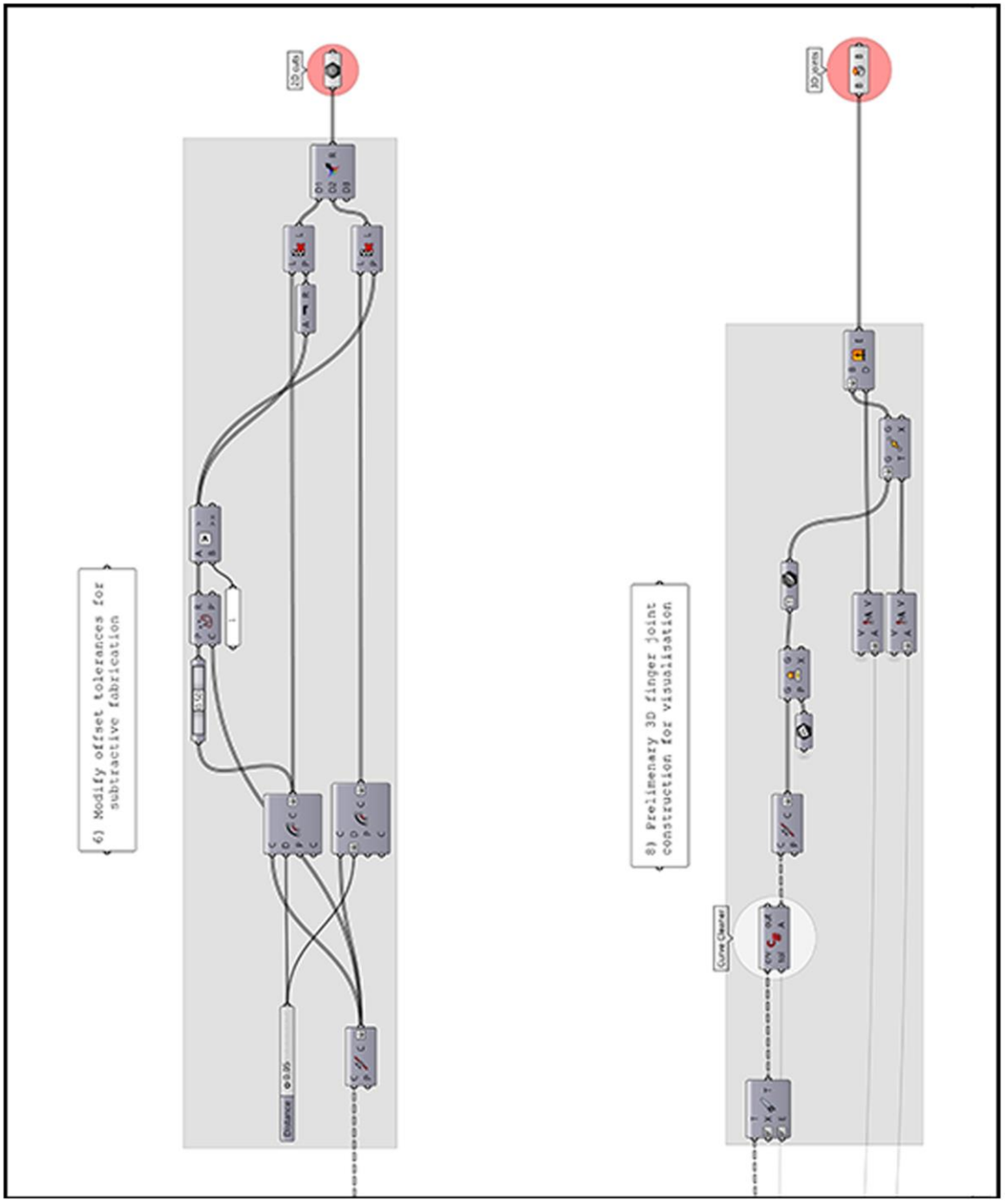
Finger joints Grasshopper script part 2:



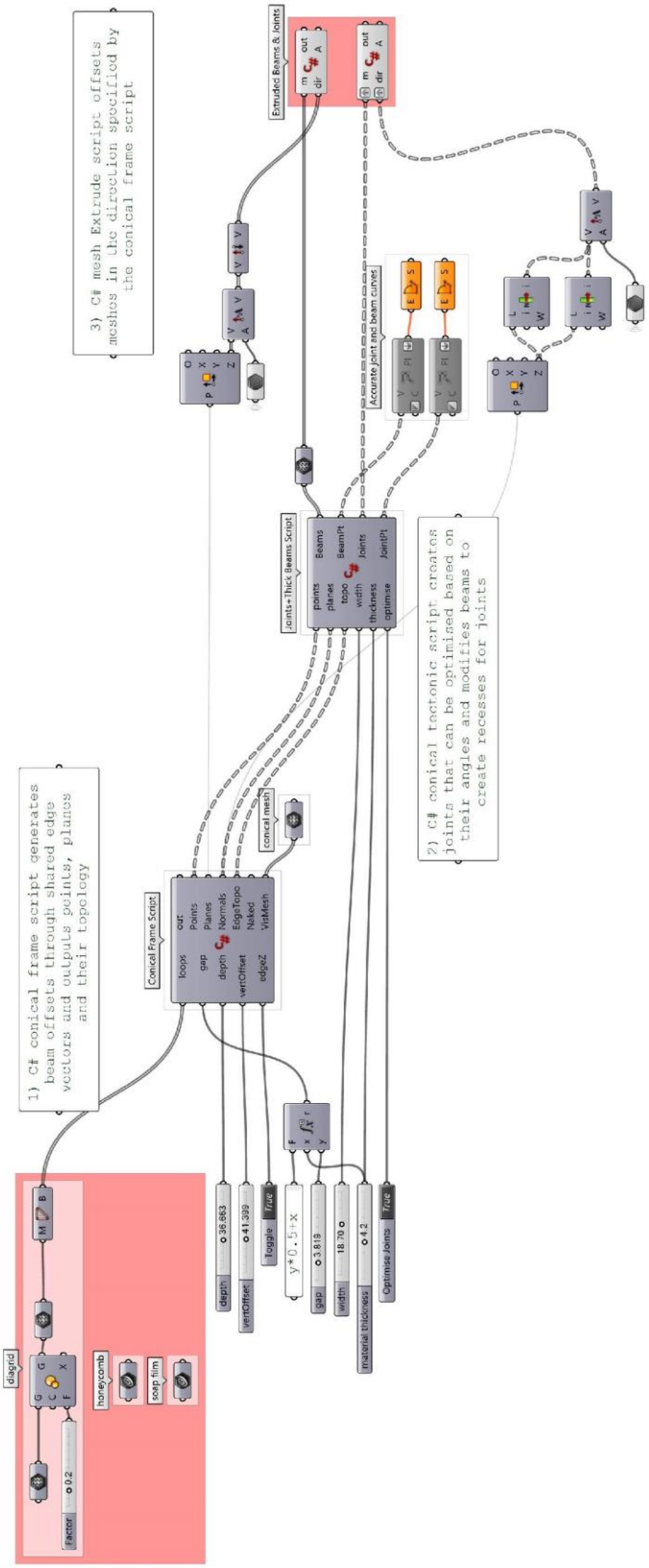
Finger joints Grasshopper script part 3:



Finger joints Grasshopper script part 4:



Conical mesh tectonics script developed for kinetic form project:



Conical frame C# script:

```
  private void RunScript(List<Polyline> loops, double gap, double depth, double
vertOffset,
        bool edgeZ, ref object Points, ref object Planes, ref object Normals, ref
object EdgeTopo,
        ref object Naked, ref object VisMesh)
  {
    lines = new List<Line>();
    segs = new List<Seg>();
    Nplanes = new List<Plane>();
    VertPlanes = new DataTree<Plane>();
    naked = new List<Line>(); //naked edges
    edgeTopo = new DataTree<int>();
    vis = new List<Mesh>();

    //create data structure for loops::
    int amount = 0;
    for (int p = 0; p < loops.Count(); p++)
    {
      Line[] segments = loops[p].GetSegments();
      int count = 0;

      for (int s = 0; s < segments.Count(); s++)
      {
        Seg S = new Seg();
        S.count = count;
        S.line = amount;
        S.loop = p;
        S.size = segments.Count();

        lines.Add(segments[s]);
        segs.Add(S);
        count++;
        amount++;
      }
    }

    //key functions::
    labelPlanes(depth);
    scaffolds(gap, edgeZ); //edgeZ switches orientation of edge planes
    buildPoints(depth, vertOffset);
    meshVis();

    //outputs:
    List<Plane> EPlanes = new List<Plane>();
    DataTree<Point3d> pts = new DataTree<Point3d>();

    for (int d = 0; d < lines.Count(); d++)
    {
      EPlanes.Add(segs[d].plane);
      GH_Path branch = new GH_Path(d);

      for (int p = 0; p < segs[d].points.Count(); p++)
      {
        pts.Add(segs[d].points[p], branch);
      }
    }
```

```
  Points = pts;
  Planes = EPlanes;
  Normals = VertPlanes;
  EdgeTopo = edgeTopo;
  Naked = naked;
  VisMesh = vis;
}

// <Custom additional code>
List<Line> lines;
List<Seg> segs;
List<Plane> Nplanes;
DataTree<Plane> VertPlanes;
List<Line> naked;
DataTree<int> edgeTopo;     //edge topology output
List<Mesh> vis;             //visualisation mesh
double tol = Rhino.RhinoDoc.ActiveDoc.ModelAbsoluteTolerance;

public class Seg
{
  public int loop;            //loop index
  public int line;            //line index
  public int count;           //current count in loop
  public int size;            //size of loop
  public int normalEnd;       //reference to vert plane at end
  public int normalStart;     //reference to vert plane at start
  public Plane plane;         //plane for current edge
  public List<Point3d> points;//points for beam
}

//mesh vis
void meshVis()
{

  for (int s = 0; s < segs.Count(); s++)
  {
    Mesh m = new Mesh();
    if(segs[s].points.Count() == 4)
    {
      m.Vertices.Add(segs[s].points[0]);
      m.Vertices.Add(segs[s].points[1]);
      m.Vertices.Add(segs[s].points[2]);
      m.Vertices.Add(segs[s].points[3]);
      m.Faces.AddFace(0, 1, 2, 3);
    }else
    {
      m.Vertices.Add(segs[s].points[0]);m.Vertices.Add(segs[s].points[1]);
      m.Vertices.Add(segs[s].points[2]);m.Vertices.Add(segs[s].points[3]);
      m.Vertices.Add(segs[s].points[4]);m.Vertices.Add(segs[s].points[5]);
      m.Vertices.Add(segs[s].points[6]);m.Vertices.Add(segs[s].points[7]);
      m.Faces.AddFace(0, 1, 6, 7);
      m.Faces.AddFace(1, 2, 5, 6);
      m.Faces.AddFace(2, 3, 4, 5);
    }
    vis.Add(m);
  }
}
```

```
//generate points
void buildPoints(double depth, double vertgap)
{
  for (int e = 0; e < segs.Count(); e++)
  {
    GH_Path branch = new GH_Path(e);
    Plane plS = Nplanes[segs[e].normalStart];
    Plane plE = Nplanes[segs[e].normalEnd];
    segs[e].points = new List<Point3d>();

    Plane aboveS = plS;
    Plane underS = plS;
    Plane aboveE = plE;
    Plane underE = plE;
    List<Point3d> pts = new List<Point3d>();
    Plane nextPl = new Plane();
    Plane lastPl = new Plane();

    //find plane on the next and last edge:
    if(segs[e].count == segs[e].size - 1)
    {
      nextPl = segs[e - segs[e].count].plane;
    }else{
      nextPl = segs[e + 1].plane;
    }
    if(segs[e].count == 0)
    {
      lastPl = segs[e + segs[e].size - 1].plane;
    }else{
      lastPl = segs[e - 1].plane;
    }

    Point3d intA,intB,intC,intD;

    //vert gap vectors:
    Vector3d back = new Vector3d();
    Vector3d forward = new Vector3d();
    back = lines[segs[e].line].From - lines[segs[e].line].To;
    forward = lines[segs[e].line].To - lines[segs[e].line].From;

    back = back - (back * plE.ZAxis) * plE.ZAxis;
    forward = forward - (forward * plS.ZAxis) * plS.ZAxis;
    back.Unitize();
    forward.Unitize();
    back = back * vertgap;
    forward = forward * vertgap;

    if(depth != 0)
    {
      //gap normal planes:
      aboveS.Translate(aboveS.ZAxis * depth * 0.5);
      underS.Translate(-underS.ZAxis * depth * 0.5);
      aboveE.Translate(aboveE.ZAxis * depth * 0.5);
      underE.Translate(-underE.ZAxis * depth * 0.5);

      Rhino.Geometry.Intersect.Intersection.PlanePlanePlane(aboveS, segs[e].plane,
```

```
lastPl, out intA);
        Rhino.Geometry.Intersect.Intersection.PlanePlanePlane(aboveE, segs[e].plane,
nextPl, out intB);
        segs[e].points.Add(intA);
        segs[e].points.Add(intB);

        if(vertgap > 0)
        {
          segs[e].points.Insert(1, intA + forward);
          segs[e].points.Insert(2, intB + back);
        }
      }

      Rhino.Geometry.Intersect.Intersection.PlanePlanePlane(underE, segs[e].plane,
nextPl, out intC);
      Rhino.Geometry.Intersect.Intersection.PlanePlanePlane(underS, segs[e].plane,
lastPl, out intD);
      segs[e].points.Insert(0, intC);
      segs[e].points.Insert(1, intD);

      if(vertgap > 0)
      {
        segs[e].points.Insert(1, intC + back);
        segs[e].points.Insert(2, intD + forward);
      }

      //ensure planarity:
      for (int p = 0; p < segs[e].points.Count(); p++)
      {
        Transform tr = Transform.PlanarProjection(segs[e].plane);
        Point3d temp = new Point3d();
        temp = segs[e].points[p];
        temp.Transform(tr);
        segs[e].points[p] = temp;
      }
    }
  }

  //generate edge planes
  void scaffolds(double gap, bool edgeZ)
  {
    for (int s = 0; s < segs.Count(); s++)
    {
      Line lineA = lines[segs[s].line];
      bool check = true;
      bool ground = false;

      //Parse naked edges
      for (int t = 0; t < segs.Count(); t++)
      {
        if(s == t)continue;
        Line lineB = lines[segs[t].line];

        if(lineA.PointAt(0.5).DistanceTo(lineB.PointAt(0.5)) < tol) check = false;
      }

      //Parse grounded edges
```

```
      if(lineA.PointAt(0.5).Z <= tol)ground = true;

      //Construct planes
      Plane XY = new Plane();
      if(check) //naked
      {
        Vector3d AxisY = (Nplanes[segs[s].normalStart].ZAxis +
Nplanes[segs[s].normalEnd].ZAxis) * 0.5;
        if(edgeZ & !ground)AxisY = new Vector3d(0, 0, 1);;
        Vector3d AxisX = lines[s].To - lines[s].From;
        XY = new Plane(lines[segs[s].line].PointAt(0.5), AxisX, AxisY);
        XY.Translate(XY.ZAxis * gap);

        naked.Add(lines[s]);  //naked lines output
      }else
      {
        Vector3d AxisY = (Nplanes[segs[s].normalStart].ZAxis +
Nplanes[segs[s].normalEnd].ZAxis) * 0.5;
        Vector3d AxisX = lines[s].To - lines[s].From;
        XY = new Plane(lines[segs[s].line].PointAt(0.5), AxisX, AxisY);
        XY.Translate(XY.ZAxis * gap);
      }
      segs[s].plane = XY;
    }
  }

  //create vert planes and label
  void labelPlanes(double depth)
  {
    List<Seg> temp = new List<Seg>();
    temp = segs;

    //get topo verts::
    List<Point3d> pts = new List<Point3d>();
    foreach (Line ln in lines)
    {
      pts.Add(ln.To);
    }

    List<Point3d> clean = new List<Point3d>();
    clean = pts;
    for (int p = 0; p < clean.Count(); p++)
    {
      for (int q = 0; q < clean.Count(); q++)
      {
        if(p == q)continue;
        if(clean[p].DistanceTo(clean[q]) < tol)
        {
          clean.RemoveAt(q);
          q--;
        }
      }
    }

    //create planes:
    for (int v = 0; v < clean.Count(); v++)
    {
```

```
      GH_Path branch = new GH_Path(v);

      for (int e = 0; e < lines.Count(); e++)
      {
        if(lines[e].From.DistanceTo(clean[v]) < tol)
        {
          segs[e].normalStart = v;
          edgeTopo.Add(e, branch);
        }
        if(lines[e].To.DistanceTo(clean[v]) < tol)
        {
          segs[e].normalEnd = v;
          edgeTopo.Add(e, branch);
        }
      }
      //output planes::
      Nplanes.Add(new Plane(clean[v], normal(clean[v])));
      Plane up = new Plane(clean[v], normal(clean[v]));
      Plane down = new Plane(clean[v], normal(clean[v]));
      up.Translate(up.ZAxis * depth * 0.5);
      down.Translate(-down.ZAxis * depth * 0.5);

      up.Flip();
      VertPlanes.Add(up, branch);
      VertPlanes.Add(down, branch);
    }
  }

  //find loop normal at a point:
  Vector3d normal(Point3d point)
  {
    Vector3d normal = new Vector3d(0, 0, 0);

    for (int i = 0; i < lines.Count(); i++)
    {
      if(lines[i].To.DistanceTo(point) < tol)
      {
        Vector3d tempA = lines[i].From - lines[i].To;
        Vector3d tempB;
        if(segs[i].count == segs[i].size - 1)
        {
          tempB = lines[i - segs[i].count].To - lines[i - segs[i].count].From;
        }else{
          tempB = lines[i + 1].To - lines[i + 1].From;
        }
        normal = normal + Vector3d.CrossProduct(tempA, tempB);
      }
    }
    normal.Unitize();
    return normal;
  }

  // </Custom additional code>
}
```

Joints+ Thickened Beams C# script:

```
                        Joints + Thickened Beams Script.txt
  private void RunScript(DataTree<Point3d> points, DataTree<Plane> planes,
DataTree<int> topo,
        double width, double thickness, bool optimise, ref object Beams, ref object
BeamPt,
        ref object Joints, ref object JointPt)
  {
    beams = new List<Beam>();
    planesTop = new List<Plane>();
    planesBottom = new List<Plane>();
    DataTree<Point3d> jointPt = new DataTree<Point3d>();
    DataTree<Mesh> joints = new DataTree<Mesh>();


    //fill planes
    for (int p = 0; p < planes.BranchCount; p++)
    {
      planesTop.Add(planes.Branch(p)[0]);
      planesBottom.Add(planes.Branch(p)[1]);
    }

    //fill beam points and refs::
    for (int i = 0; i < points.BranchCount; i++)
    {
      Beam beam = new Beam();
      beam.points = new List<Point3d>();
      beam.points = points.Branch(i);
      beam.beam = i;
      beams.Add(beam);
    }

    //fill verts and key points::
    for (int v = 0; v < planes.BranchCount; v++)//go through verts
    {
      List<Trio> preJoints = new List<Trio>();
      sortOut = new List<Trio>();
      GH_Path branch = new GH_Path(v);

      //calculate vector offsets
      for (int t = 0; t < topo.Branch(v).Count(); t++)//go through topos
      {
        int Ast,Aet,Asb,Aeb;
        Vector3d dir =
          sortPoints(beams[topo.Branch(v)[t]].points, planes.Branch(v)[0], out Ast,
out Aet);
        if(optimise == true)
        {
          double ang = double.PositiveInfinity;
          for (int u = 0; u < topo.Branch(v).Count(); u++)//go through topos
          {
            Vector3d temp = sortPoints(beams[topo.Branch(v)[u]].points,
planes.Branch(v)[1], out Asb, out Aeb);
            if(Vector3d.VectorAngle(dir, temp) < ang && Vector3d.VectorAngle(dir,
temp) > tol)
            {
              ang = Vector3d.VectorAngle(dir, temp);
            }
          }
```

```
        ang *= 0.5;
        double opp = width * 0.5;
        ang = opp / Math.Tan(ang);
        dir *= ang;
      }else{
        dir *= width;
      }

      //sort beam classes relative to plane pointers:
      if(  beams[topo.Branch(v)[t]].vertA < 0)
      {
        beams[topo.Branch(v)[t]].vecA = dir;

        sortPoints(beams[topo.Branch(v)[t]].points, planes.Branch(v)[1], out Asb,
out Aeb);

        beams[topo.Branch(v)[t]].Ast = Ast;
        beams[topo.Branch(v)[t]].Aet = Aet;
        beams[topo.Branch(v)[t]].Asb = Asb;
        beams[topo.Branch(v)[t]].Aeb = Aeb;
        beams[topo.Branch(v)[t]].vertA = v;

      }else{
        beams[topo.Branch(v)[t]].vecB = dir;
        sortPoints(beams[topo.Branch(v)[t]].points, planes.Branch(v)[0], out Ast,
out Aet);
        sortPoints(beams[topo.Branch(v)[t]].points, planes.Branch(v)[1], out Asb,
out Aeb);

        beams[topo.Branch(v)[t]].Bst = Ast;
        beams[topo.Branch(v)[t]].Bet = Aet;
        beams[topo.Branch(v)[t]].Bsb = Asb;
        beams[topo.Branch(v)[t]].Beb = Aeb;
        beams[topo.Branch(v)[t]].vertB = v;
      }

      Trio trio = new Trio();
      extraPoints(beams[topo.Branch(v)[t]], v, thickness, out trio);
      preJoints.Add(trio);
    }

    //Create joints::
    sort(planesTop[v], preJoints);

    Mesh mesh = new Mesh();
    jointPt.AddRange(buildJoints(sortOut, v, out mesh, true), branch);
    joints.Add(mesh, branch);
    jointPt.AddRange(buildJoints(sortOut, v, out mesh, false), branch);
    joints.Add(mesh, branch);
  }

  DataTree<Point3d> beamPoints = new DataTree<Point3d>(); //beam points for output
  List<Mesh> beamVis = new List<Mesh>();  //beam meshes for output

  //go through beams
  for (int b = 0; b < beams.Count(); b++)
  {
```

```
    GH_Path branch = new GH_Path(b);

    Mesh mesh = new Mesh();
    beamPoints.AddRange(buildBeams(beams[b], out mesh, width, thickness), branch);
    beamVis.Add(mesh);
  }
  Joints = joints;
  JointPt = jointPt;
  Beams = beamVis;
  BeamPt = beamPoints;
}

// <Custom additional code>
double tol = Rhino.RhinoDoc.ActiveDoc.ModelAbsoluteTolerance;
List<Beam> beams;
List<Plane> planesTop;
List<Plane> planesBottom;

class Beam
{
  public List<Point3d>points;
  public int beam;
  public int vertA = -1;
  public int vertB = -1;

  //Point refs::
  public int Ast; //A: side A, s: at start, t: top
  public int Aet;
  public int Asb;
  public int Aeb;
  public int Bst;
  public int Bet;
  public int Bsb;
  public int Beb; //B: side B, e: at end, b: bottom

  //Extra points::
  public Point3d Ae0; //extra points in clockwise direction
  public Point3d Ae1;
  public Point3d Ae2;
  public Point3d Ae3;
  public Point3d Ae4;
  public Point3d Ae5;
  public Point3d Be0; //extra points for side B
  public Point3d Be1;
  public Point3d Be2;
  public Point3d Be3;
  public Point3d Be4;
  public Point3d Be5;

  //vector ref::
  public Vector3d vecA;
  public Vector3d vecB;
}

class Trio  //class for joints
{
  public Point3d startT;
```

```
  public Point3d endT;
  public Point3d midT;

  public Point3d startB;
  public Point3d endB;
  public Point3d midB;
}

//Construct Joints
List<Point3d> buildJoints(List<Trio> trio, int plane, out Mesh mesh, bool top)
{
  List<Point3d> points = new List<Point3d>();
  Mesh M = new Mesh();
  Mesh patch = new Mesh();
  int count = 0;
  List<Point3d> pts = new List<Point3d>();  //points for patch

  if(trio[0].midT.DistanceTo(trio[1].midT) > tol
    && trio[0].startT.DistanceTo(trio[1].startT) < tol)
  {
    trio.Add(trio[0]);
    trio.RemoveAt(0);
  }

  for (int t = 0; t < trio.Count() - 1; t += 2)
  {
    Trio now = trio[t];
    Trio next = trio[t + 1];

    Point3d end = now.endT;
    Point3d mid = now.midT;
    Point3d start = now.startT;
    Point3d endN = next.endT;
    Point3d midN = next.midT;
    Point3d startN = next.startT;
    Point3d pln = planesTop[plane].Origin;

    if(top)
    {
      end = now.endB;
      mid = now.midB;
      start = now.startB;
      endN = next.endB;
      midN = next.midB;
      startN = next.startB;
      pln = planesBottom[plane].Origin;
    }

    if(t != trio.Count() - 1)
    {
      if(mid.DistanceTo(midN) < tol)
      {
        count += 4;
        //points::
        points.Add(end);
        points.Add(mid);
        points.Add(endN);
```

```
        points.Add(startN);
        //mesh arms::
        M.Vertices.Add(start);
        M.Vertices.Add(end);
        M.Vertices.Add(endN);
        M.Vertices.Add(startN);
        M.Faces.AddFace(count - 4, count - 3, count - 2, count - 1);
        pts.Add(start);
        //pts.Add(next.startT);
      }else{
        count += 8;
        //points::
        points.Add(end);
        points.Add(mid);
        points.Add(pln);
        points.Add(midN);
        points.Add(endN);
        points.Add(startN);
        //mesh arms::
        M.Vertices.Add(start);
        M.Vertices.Add(end);
        M.Vertices.Add(mid);
        M.Vertices.Add(pln);
        M.Vertices.Add(pln);
        M.Vertices.Add(midN);
        M.Vertices.Add(endN);
        M.Vertices.Add(startN);

        M.Faces.AddFace(count - 4, count - 3, count - 2, count - 1);
        M.Faces.AddFace(count - 8, count - 7, count - 6, count - 5);
        pts.Add(start);
        pts.Add(pln);
      }
    }
  }

  pts.Reverse();
  patch = Meshpatch(pts);
  patch.Append(M);

  mesh = patch;
  return points;
}

//Construct beams
List<Point3d> buildBeams(Beam beam, out Mesh mesh, double width, double thickness)
{
  Mesh m = new Mesh();
  List<Point3d> pts = new List<Point3d>();

  //construct point list:
  pts.Add(beam.points[beam.Aet]);        //0
  pts.Add(beam.Ae0);                     //1
  pts.Add(beam.Ae1);                     //2
  pts.Add(beam.Ae2);                     //3
  pts.Add(beam.Ae3);                     //4
  pts.Add(beam.Ae4);//debug              //5
```

```
  pts.Add(beam.Ae5);//debug              //6
  pts.Add(beam.points[beam.Aeb]);        //7
  pts.Add(beam.points[beam.Beb]);        //8
  pts.Add(beam.Be5);                     //9
  pts.Add(beam.Be4);                     //10
  pts.Add(beam.Be3);                     //11
  pts.Add(beam.Be2);                     //12
  pts.Add(beam.Be1);//debug              //13
  pts.Add(beam.Be0);//debug              //14
  pts.Add(beam.points[beam.Bet]);        //15

  //construct mesh
  m.Vertices.AddVertices(pts);
  m.Faces.AddFace(12, 13, 10, 11);
  m.Faces.AddFace(13, 15, 8, 10);
  m.Faces.AddFace(15, 0, 7, 8);
  m.Faces.AddFace(0, 2, 5, 7);
  m.Faces.AddFace(2, 3, 4, 5);
  m.Faces.AddFace(14, 15, 13);
  m.Faces.AddFace(10, 8, 9);
  m.Faces.AddFace(0, 1, 2);
  m.Faces.AddFace(5, 6, 7);

  mesh = m;
  return pts;
}

//Add extra points to beam
Beam extraPoints(Beam beam, int plane, double thickness, out Trio three)
{
  Beam m = new Beam();
  m = beam;
  Trio tri = new Trio();
  double t = 0;

  if(planesTop[plane].Origin.DistanceTo(planesTop[beam.vertA].Origin) < tol)
  {
    Line edgeA = new Line(beam.points[beam.Ast], beam.points[beam.Asb]);

    //top points::
    Line lnAtop = new Line(beam.points[beam.Ast], beam.points[beam.Aet]);
    Plane plnA = new Plane(planesTop[plane].Origin + beam.vecA, beam.vecA);
    Plane plnA2 = new Plane(beam.points[m.Ast] +
      (planesTop[plane].ZAxis * thickness), planesTop[plane].ZAxis);
    Rhino.Geometry.Intersect.Intersection.LinePlane(lnAtop, plnA, out t);
    m.Ae0 = lnAtop.PointAt(t);
    Rhino.Geometry.Intersect.Intersection.LinePlane(edgeA, plnA2, out t);
    m.Ae2 = edgeA.PointAt(t);
    tri.midT = planesTop[plane].ClosestPoint(plnA.Origin);
    tri.endT = planesTop[plane].ClosestPoint(m.Ae0);
    tri.startT = planesTop[plane].ClosestPoint(beam.points[m.Ast]);

    //bottom points::
    Line lnAbottom = new Line(beam.points[beam.Asb], beam.points[beam.Aeb]);
    Plane plnA3 = new Plane(beam.points[m.Asb] +
      (planesBottom[plane].ZAxis * thickness), planesBottom[plane].ZAxis);
    Rhino.Geometry.Intersect.Intersection.LinePlane(edgeA, plnA3, out t);
```

```
      m.Ae3 = edgeA.PointAt(t);
      Rhino.Geometry.Intersect.Intersection.LinePlane(lnAbottom, plnA, out t);
      m.Ae5 = lnAbottom.PointAt(t);
      tri.midB = planesBottom[plane].ClosestPoint(planesBottom[plane].Origin +
beam.vecA);
      tri.endB = planesBottom[plane].ClosestPoint(m.Ae5);
      tri.startB = planesBottom[plane].ClosestPoint(beam.points[m.Asb]);

      //inner beam points::
      Vector3d Adown = m.Ae5 - m.Ae0;
      Adown.Unitize();
      Adown *= thickness;
      m.Ae1 = m.Ae0 + Adown;
      Adown.Reverse();
      m.Ae4 = m.Ae5 + Adown;

    }else if(planesTop[plane].Origin.DistanceTo(planesTop[beam.vertB].Origin) < tol)
    {
      Line edgeB = new Line(beam.points[beam.Bst], beam.points[beam.Bsb]);

      //top points::
      Line lnBtop = new Line(beam.points[beam.Bst], beam.points[beam.Bet]);
      Plane plnB = new Plane(planesTop[plane].Origin + beam.vecB, beam.vecB);
      Plane plnB2 = new Plane(beam.points[m.Bst] +
        (planesTop[plane].ZAxis * thickness), planesTop[plane].ZAxis);
      Rhino.Geometry.Intersect.Intersection.LinePlane(lnBtop, plnB, out t);
      m.Be0 = lnBtop.PointAt(t);
      Rhino.Geometry.Intersect.Intersection.LinePlane(edgeB, plnB2, out t);
      m.Be2 = edgeB.PointAt(t);
      tri.midT = planesTop[plane].ClosestPoint(plnB.Origin);
      tri.endT = planesTop[plane].ClosestPoint(m.Be0);
      tri.startT = planesTop[plane].ClosestPoint(beam.points[m.Bst]);

      //bottom points::
      Line lnBbottom = new Line(beam.points[beam.Bsb], beam.points[beam.Beb]);
      Plane plnB3 = new Plane(beam.points[m.Bsb] +
        (planesBottom[plane].ZAxis * thickness), planesBottom[plane].ZAxis);
      Rhino.Geometry.Intersect.Intersection.LinePlane(edgeB, plnB3, out t);
      m.Be3 = edgeB.PointAt(t);
      Rhino.Geometry.Intersect.Intersection.LinePlane(lnBbottom, plnB, out t);
      m.Be5 = lnBbottom.PointAt(t);
      tri.midB = planesBottom[plane].ClosestPoint(planesBottom[plane].Origin +
beam.vecB);
      tri.endB = planesBottom[plane].ClosestPoint(m.Be5);
      tri.startB = planesBottom[plane].ClosestPoint(beam.points[m.Bsb]);

      //inner beam points::
      Vector3d Bdown = m.Be5 - m.Be0;
      Bdown.Unitize();
      Bdown *= thickness;
      m.Be1 = m.Be0 + Bdown;
      Bdown.Reverse();
      m.Be4 = m.Be5 + Bdown;
    }
    three = tri;
    return m;
  }
```

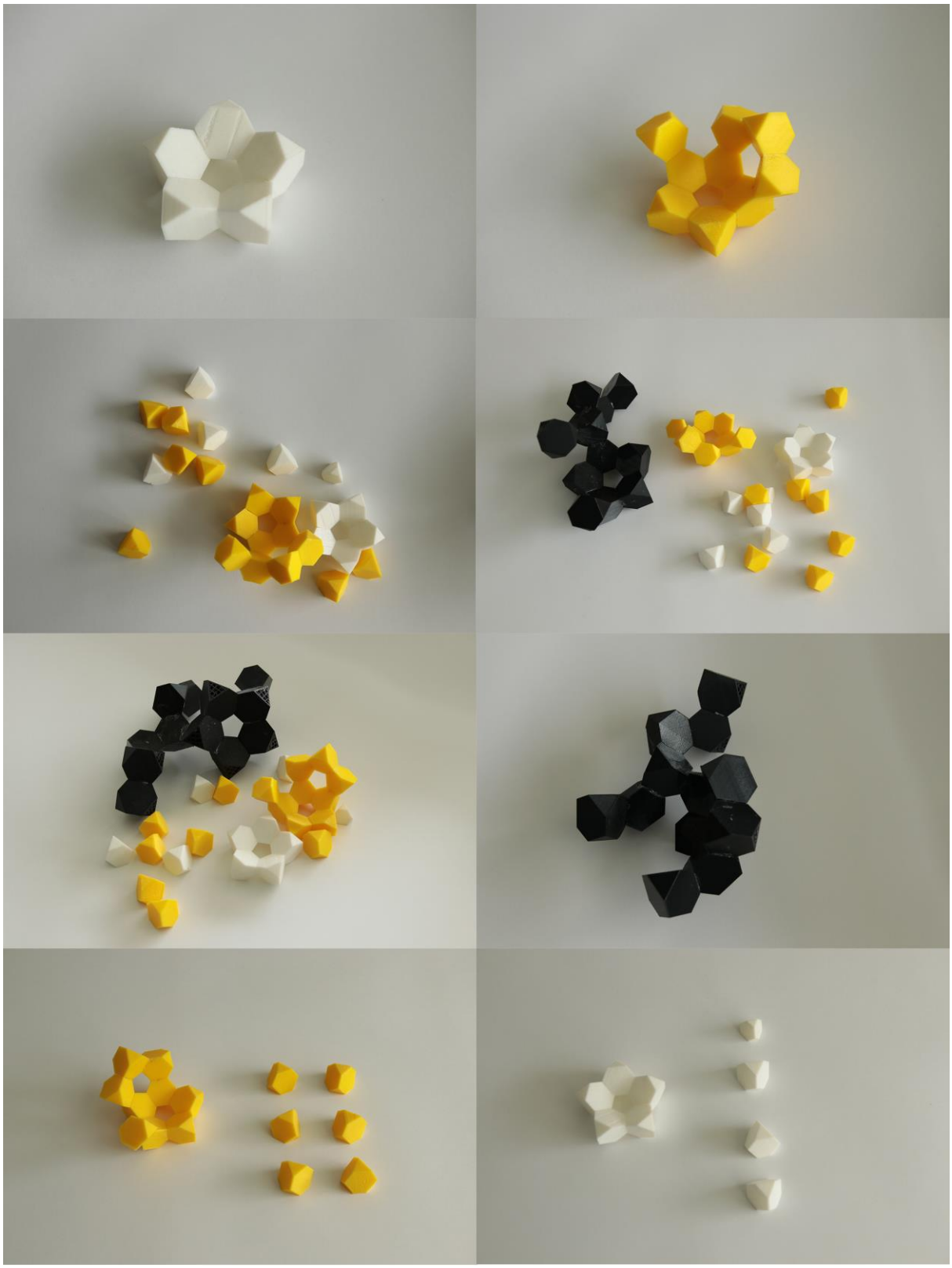
```
//Sort lines and vectors around a plane::
List<Trio>sortOut;
void sort(Plane plane, List<Trio> ListToSort)
{
  List<Trio> temp = new List<Trio>();
  temp = ListToSort;

  if(temp.Count() > 0)
  {
    double outAng = double.PositiveInfinity;
    int count = 0;

    for (int i = 0; i < temp.Count(); i++)
    {
      Vector3d tempV = temp[i].endT - plane.Origin;

      if(Vector3d.VectorAngle(tempV, plane.XAxis, plane) < outAng)
      {
        outAng = Vector3d.VectorAngle(tempV, plane.XAxis, plane);
        count = i;
      }
    }
    Trio tr = new Trio();
    tr = temp[count];
    sortOut.Add(tr);
    temp.RemoveAt(count);
    sort(plane, temp);
  }
}

//Assign vectors::
Vector3d sortPoints(List<Point3d> points, Plane plane, out int start, out int end)
{
  start = 0;
  end = 0;
  List<int> planar = new List<int>();
  List<Point3d> debug = new List<Point3d>();

  //find planar points::
  for (int p = 0; p < points.Count(); p++)
  {
    double dis = Math.Abs(plane.ClosestPoint(points[p]).DistanceTo(points[p]));
    if(dis <= tol * 1000)
    {
      planar.Add(p);
      debug.Add(points[p]);
    }
  }

  //orient points::
  double inf = double.PositiveInfinity;
  int removeAt = 0;
  for (int e = 0; e < planar.Count(); e++)
  {
    double dis = plane.Origin.DistanceTo(points[planar[e]]);
    if(dis < inf)
```

```
    {
      inf = dis;
      start = planar[e];
      removeAt = e;
    }
  }
  planar.RemoveAt(removeAt);

  inf = double.PositiveInfinity;
  for (int f = 0; f < planar.Count(); f++)
  {
    double dis = plane.Origin.DistanceTo(points[planar[f]]);
    if(dis < inf)
    {
      inf = dis;
      end = planar[f];
    }
  }

  //prepare joints curves::
  Vector3d forward = points[end] - points[start];
  forward.Unitize();
  return forward;
}

//Mesh patch::
Mesh Meshpatch(List<Point3d> points)
{
  Mesh patch = new Mesh();

  int count = 0;
  int face = points.Count();

  if(face % 2 != 0) //if odd points
  {
    patch.Vertices.Add(points[(int) (face * 0.5 - 1)]);           //0
    patch.Vertices.Add(points[(int) (face * 0.5)]);               //1
    patch.Vertices.Add(points[(int) (face * 0.5 + 1)]);           //2
    patch.Faces.AddFace(2, 1, 0);
  }
  if(face < 4)return patch;
  for (int p = 0; p < points.Count(); p++)
  {
    if(count == 1 && p < face - 1)
    {
      patch.Vertices.Add(points[p]);             //0
      patch.Vertices.Add(points[p - 1]);         //1
      patch.Vertices.Add(points[face]);          //2
      patch.Vertices.Add(points[face - 1]);      //3
      patch.Faces.AddFace(patch.Vertices.Count - 4, patch.Vertices.Count - 3,
      patch.Vertices.Count - 2, patch.Vertices.Count - 1);
      count = 0;
    }
    count++;
    face--;
  }
  return patch;
```

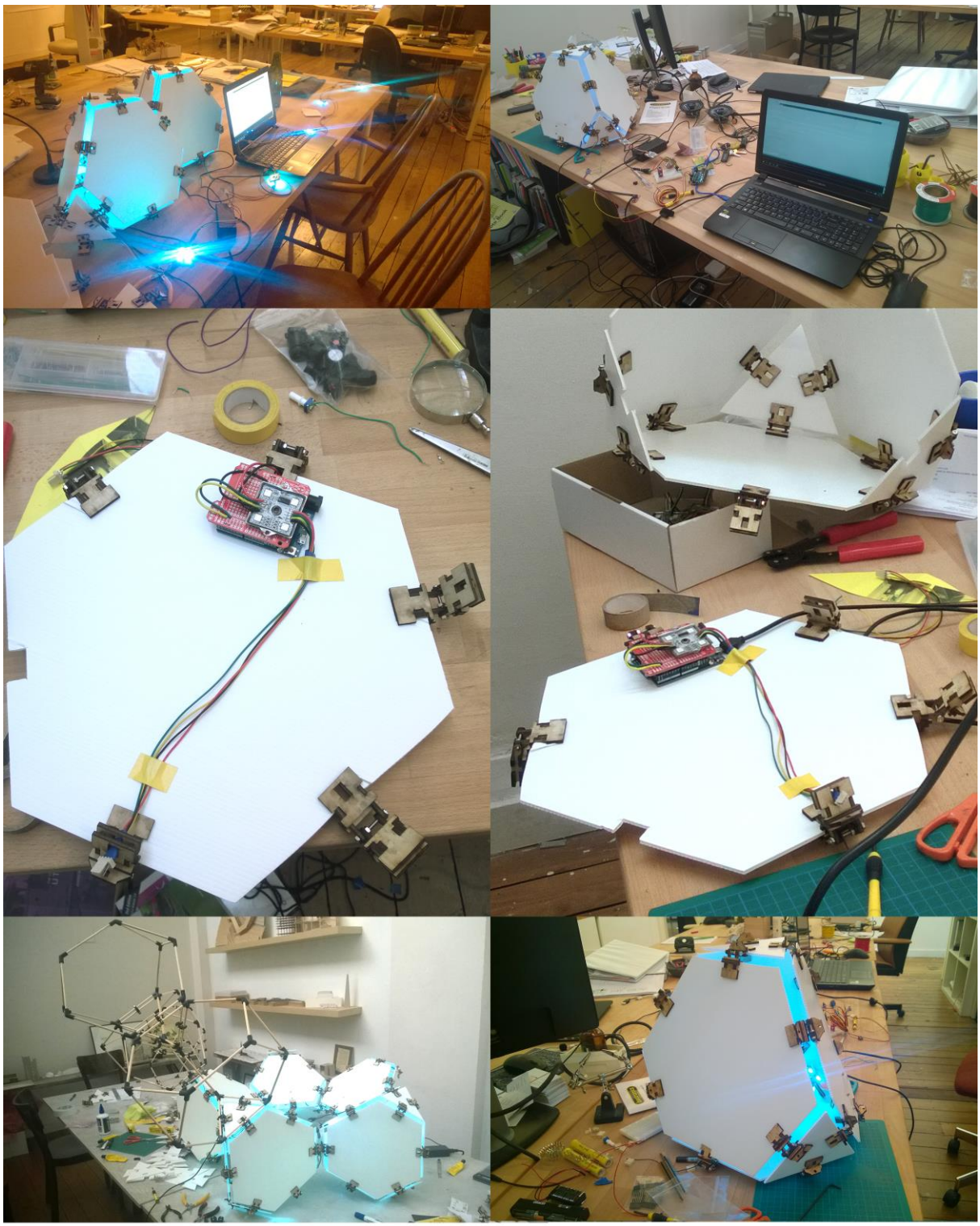
```
  }
  // </Custom additional code>
}

//extra
  Mesh ExtrudeMesh(Mesh mesh, Vector3d direction)
  {
    Mesh msh = new Mesh();
    mesh.Flip(true, true, true);
    Line[] segs = mesh.GetNakedEdges()[0].GetSegments();

    int count = 0;
    for (int s = 0; s < segs.Count(); s++)
    {
      msh.Vertices.Add(segs[s].To);
      msh.Vertices.Add(segs[s].From);
      msh.Vertices.Add(segs[s].From + direction);
      msh.Vertices.Add(segs[s].To + direction);
      msh.Faces.AddFace(count, count + 1, count + 2, count + 3);
      count += 4;
    }

    msh.Append(mesh);
    mesh.Translate(direction);
    //mesh.Flip(true, true, true);
    msh.Append(mesh);
    msh.UnifyNormals();
    return msh;
  }
```

D. Visual diary – progress images

Cellular Form- aggregation studies with 3D printed scaled pieces



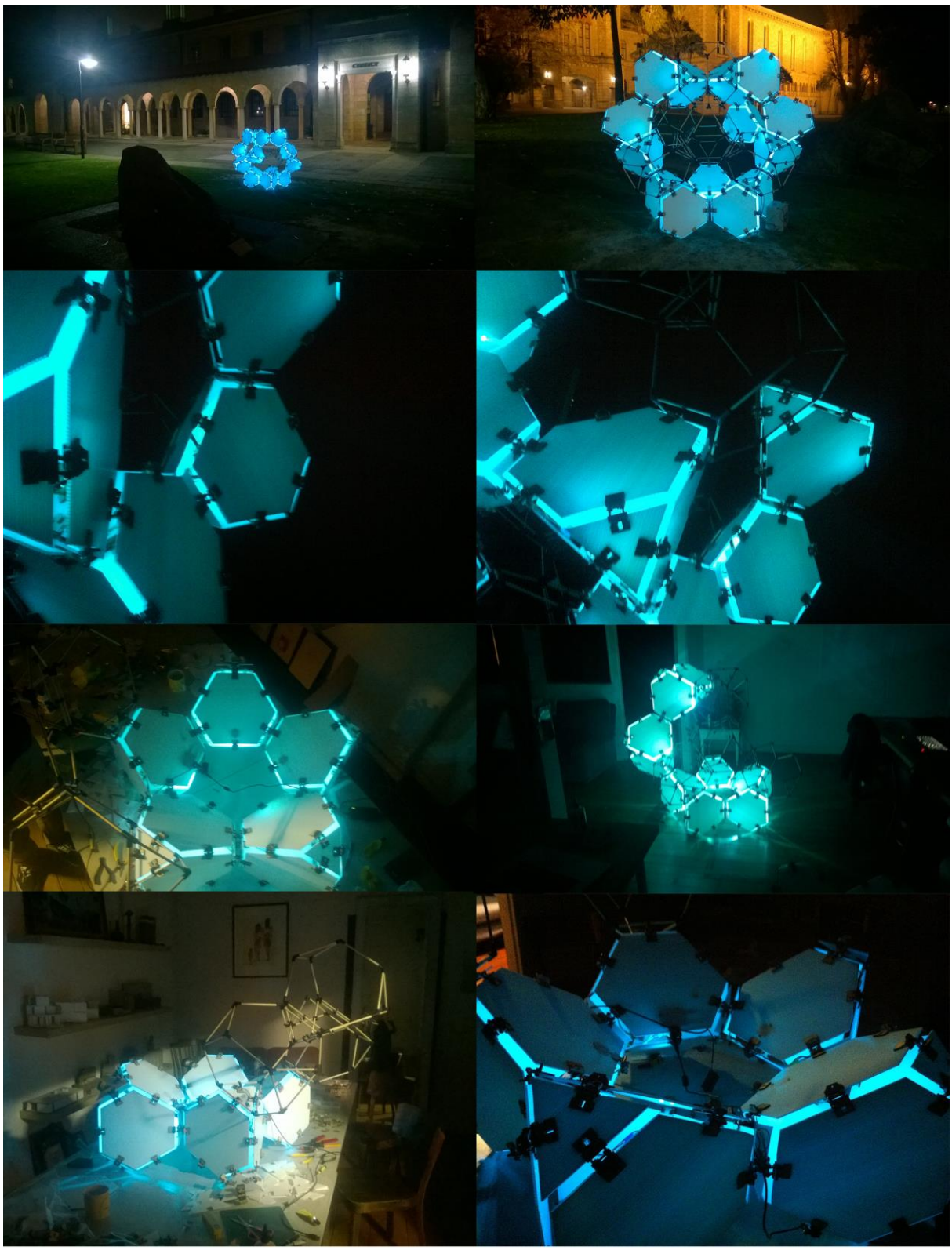
Cellular Form- embedded responsive components



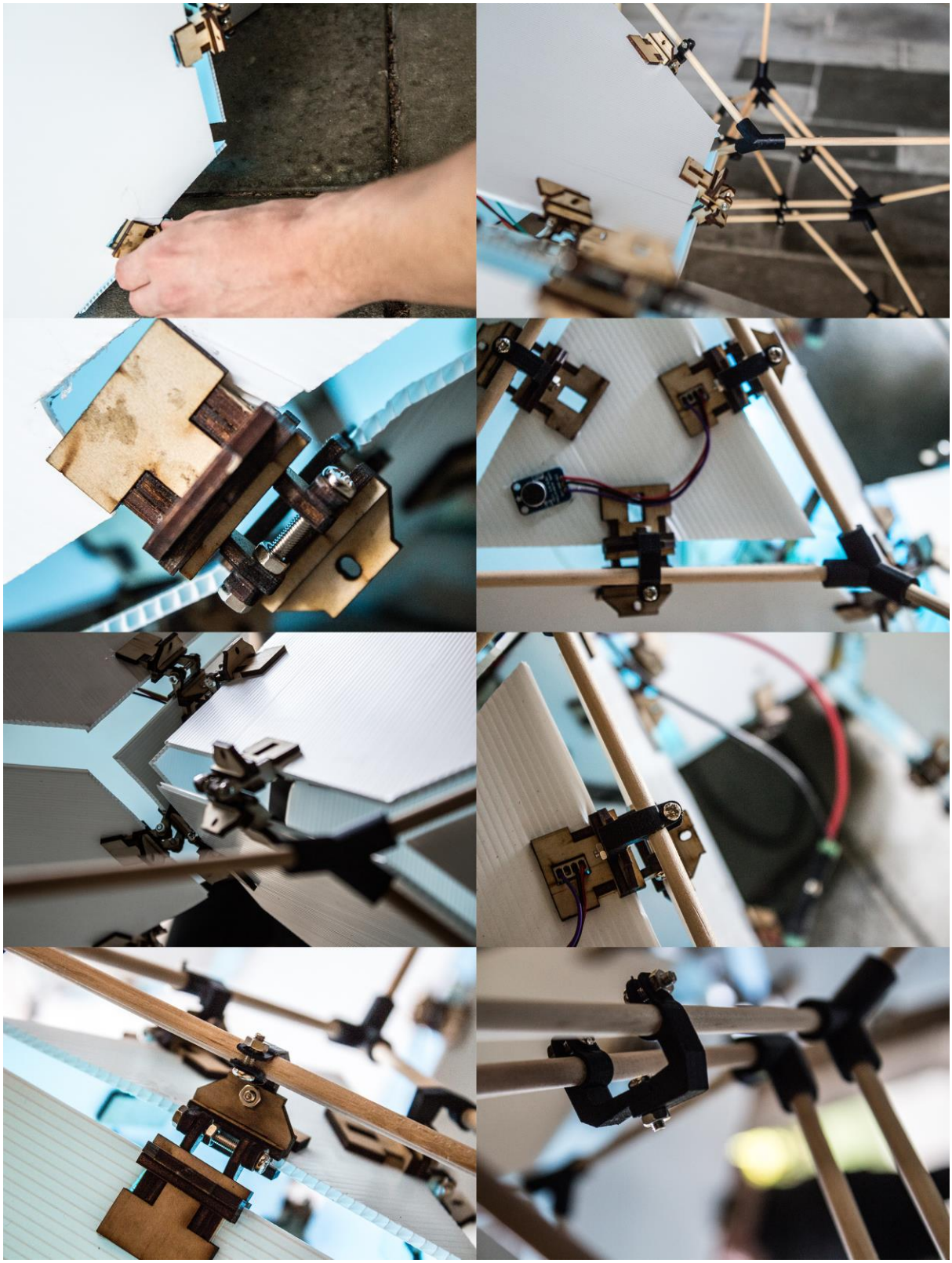
Cellular Form- fabrication of hinge joints

Cellular Form- transportation and storage

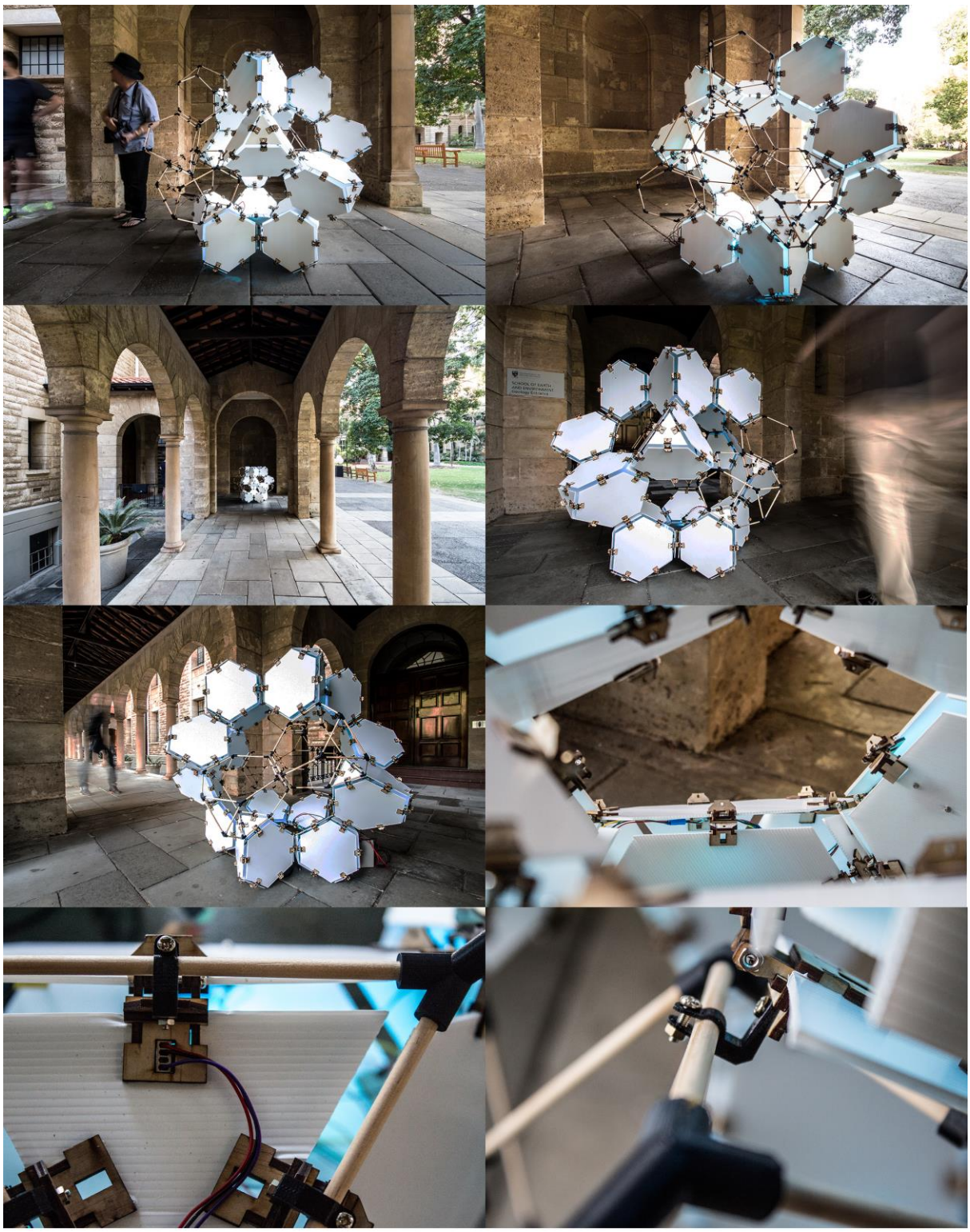
Cellular Form- light tests



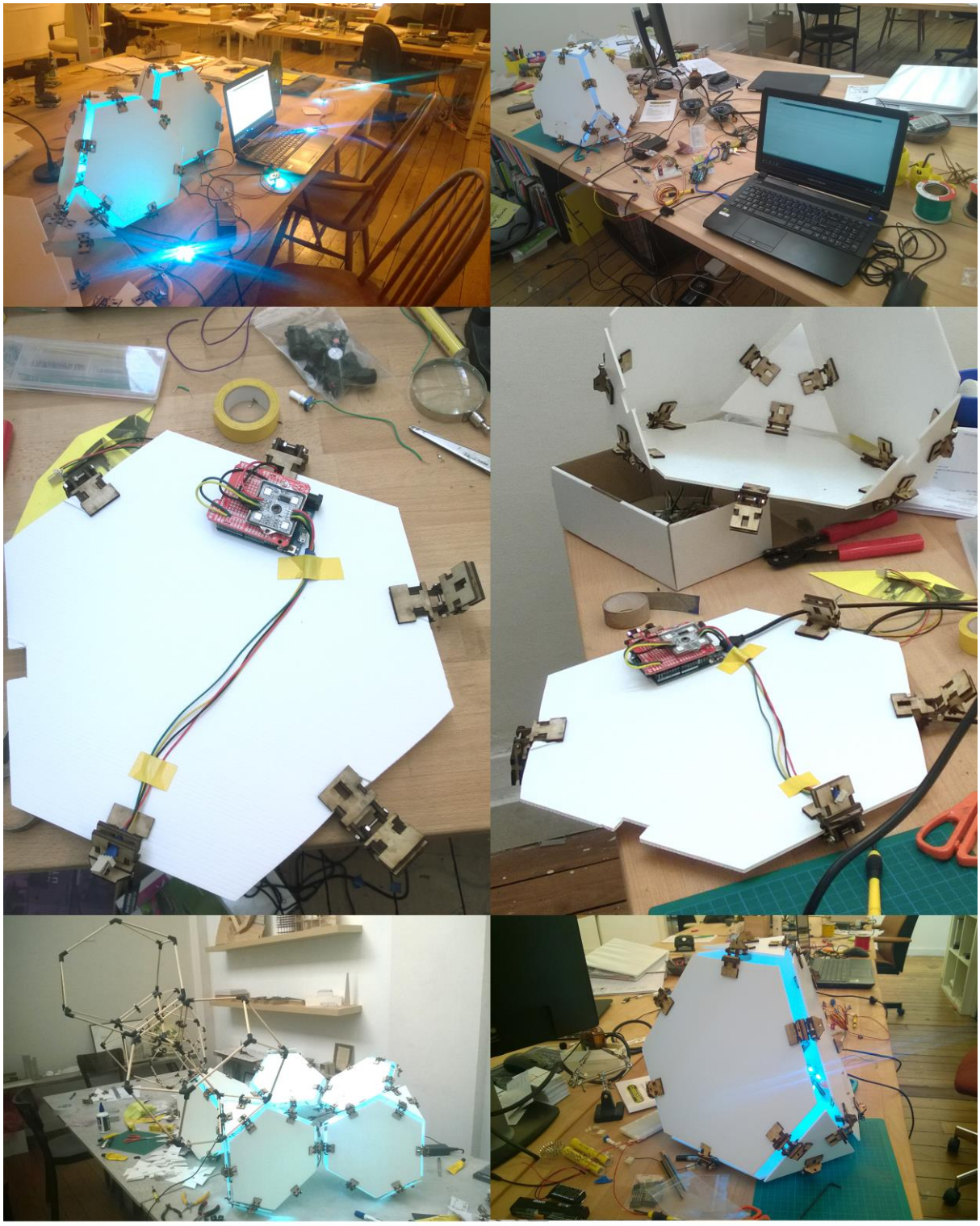
Cellular Form- tectonic details and points of failure



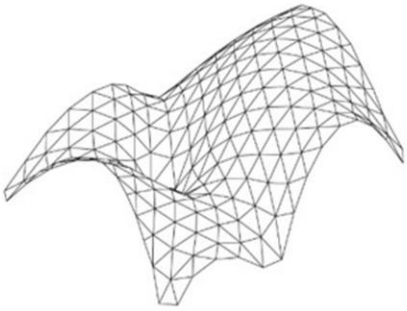
Cellular Form- public interaction



Mediated Form- early form finding experiments

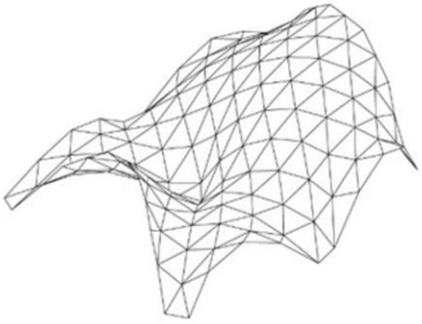


Mediated Form- assembly instructions

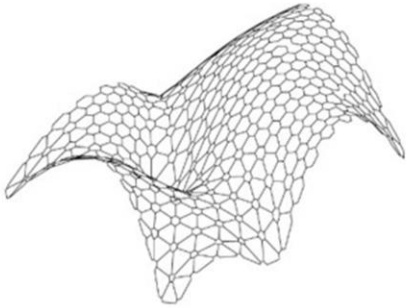


378 faces

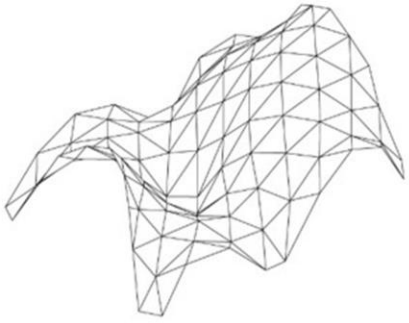
Recessed
Verticies



228 faces



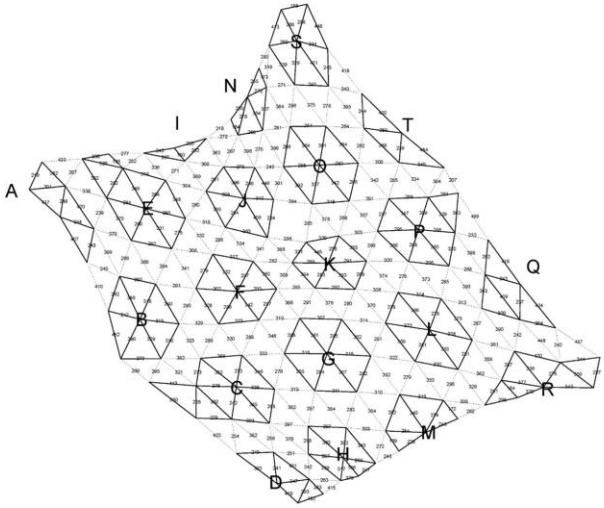
Attractor-based
gradient pattern

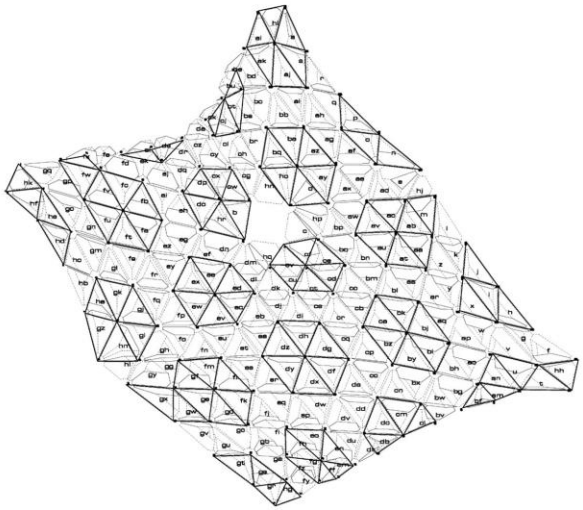


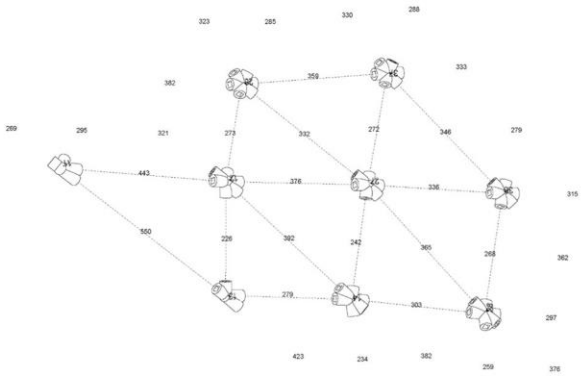
170 faces

solid components

Mediated Form- assembly instructions







Mediated Form- assembly of hexagonal parts & material selection

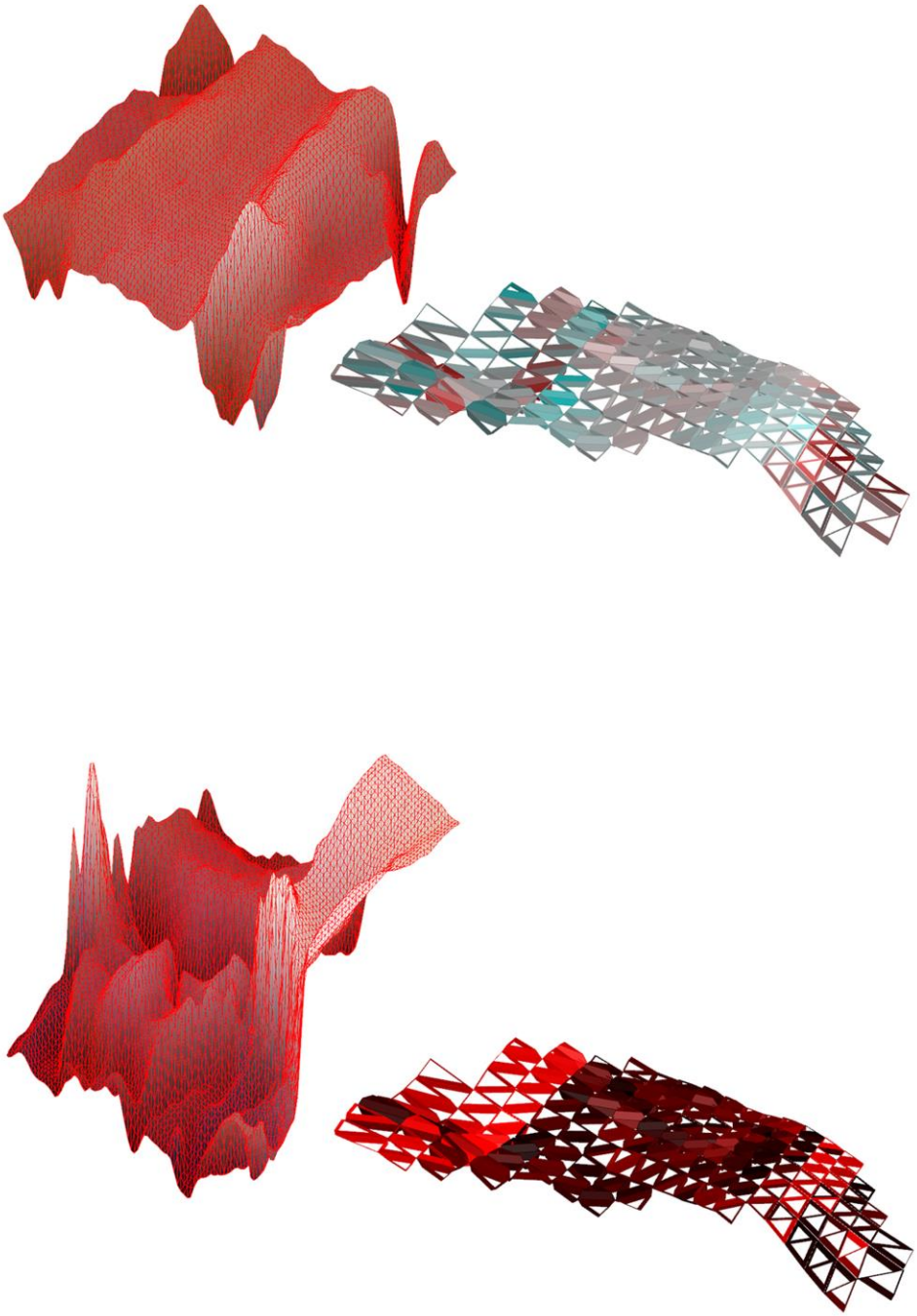
Mediated Form- assembly of large skeletal frame and cladding

Mediated Form- final assembly from three sub-parts

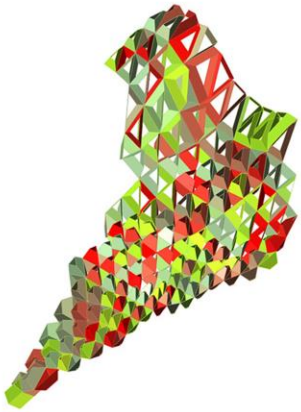
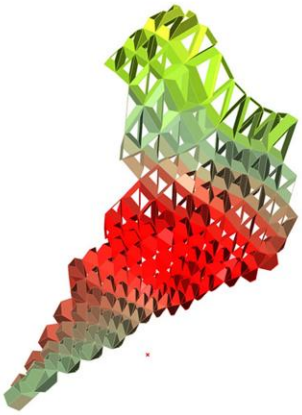
Mediated Form- on site installation and transportation

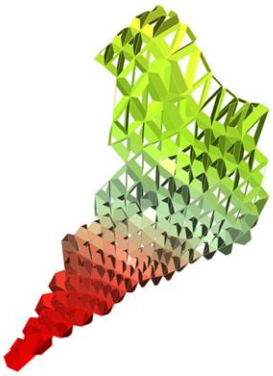
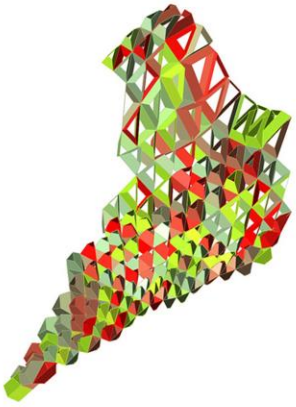
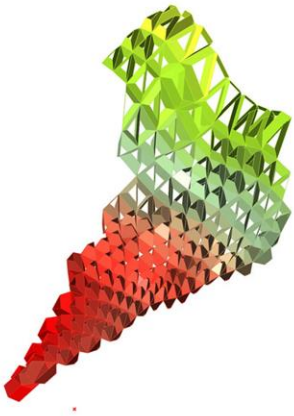
Mediated Form- public interaction

Kinetic Form- Initial simulation of interactivity through Firefly and Grasshopper

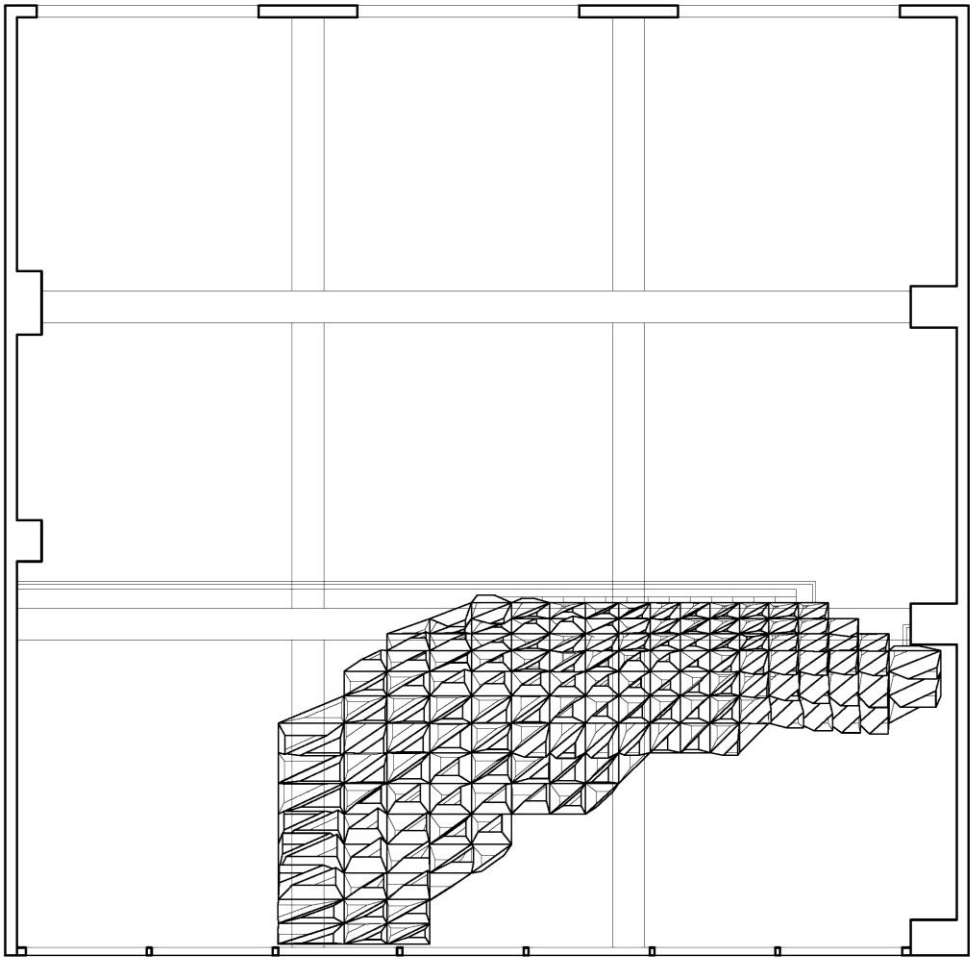


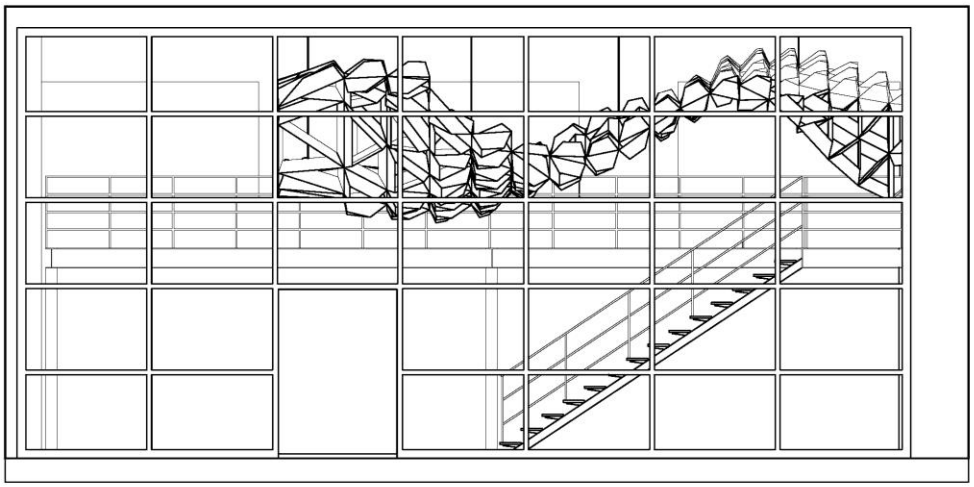
Kinetic Form- Initial simulation of interactivity variations using Firefly and Grasshopper

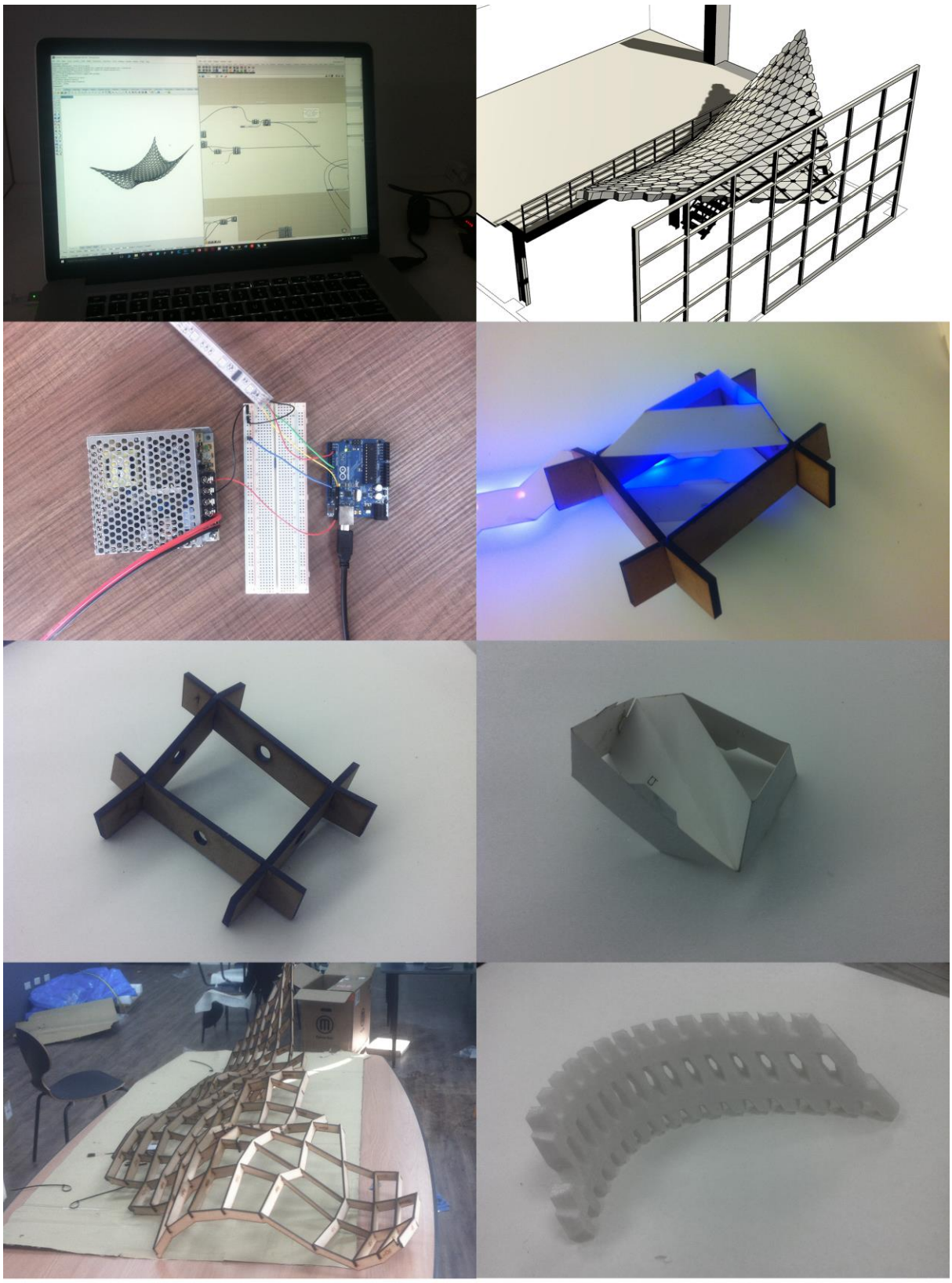




Kinetic Form- Initial proposal for Taiwan space kinetic installation

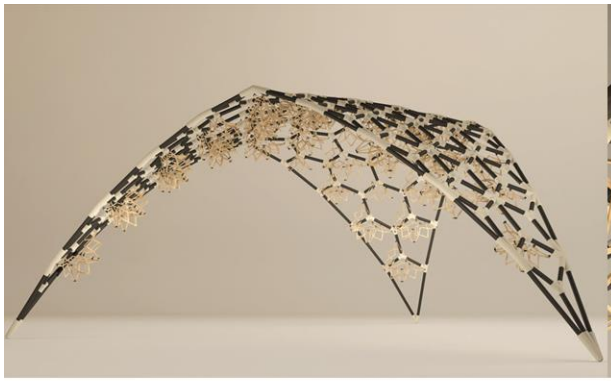


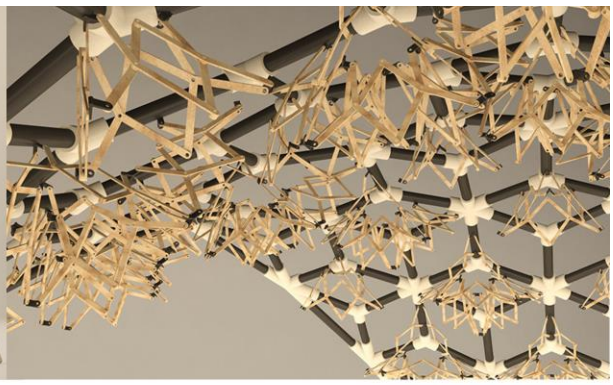


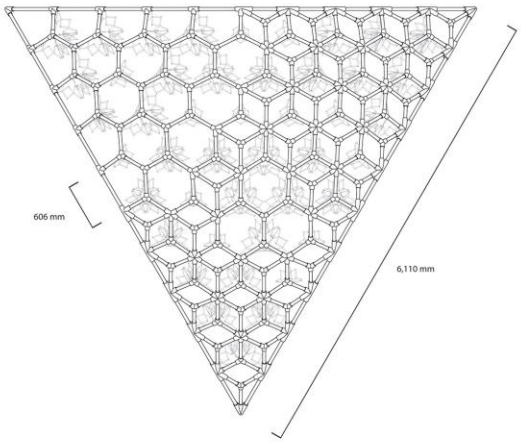
Kinetic Form- Initial interactive component tests

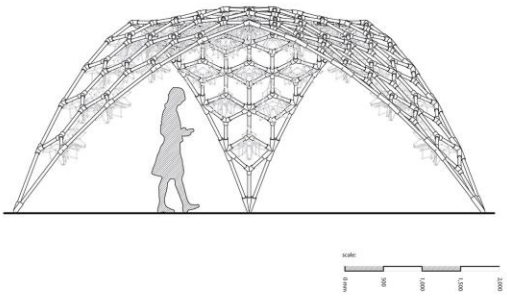


Kinetic Form- Initial carrier surface design for Shenzen



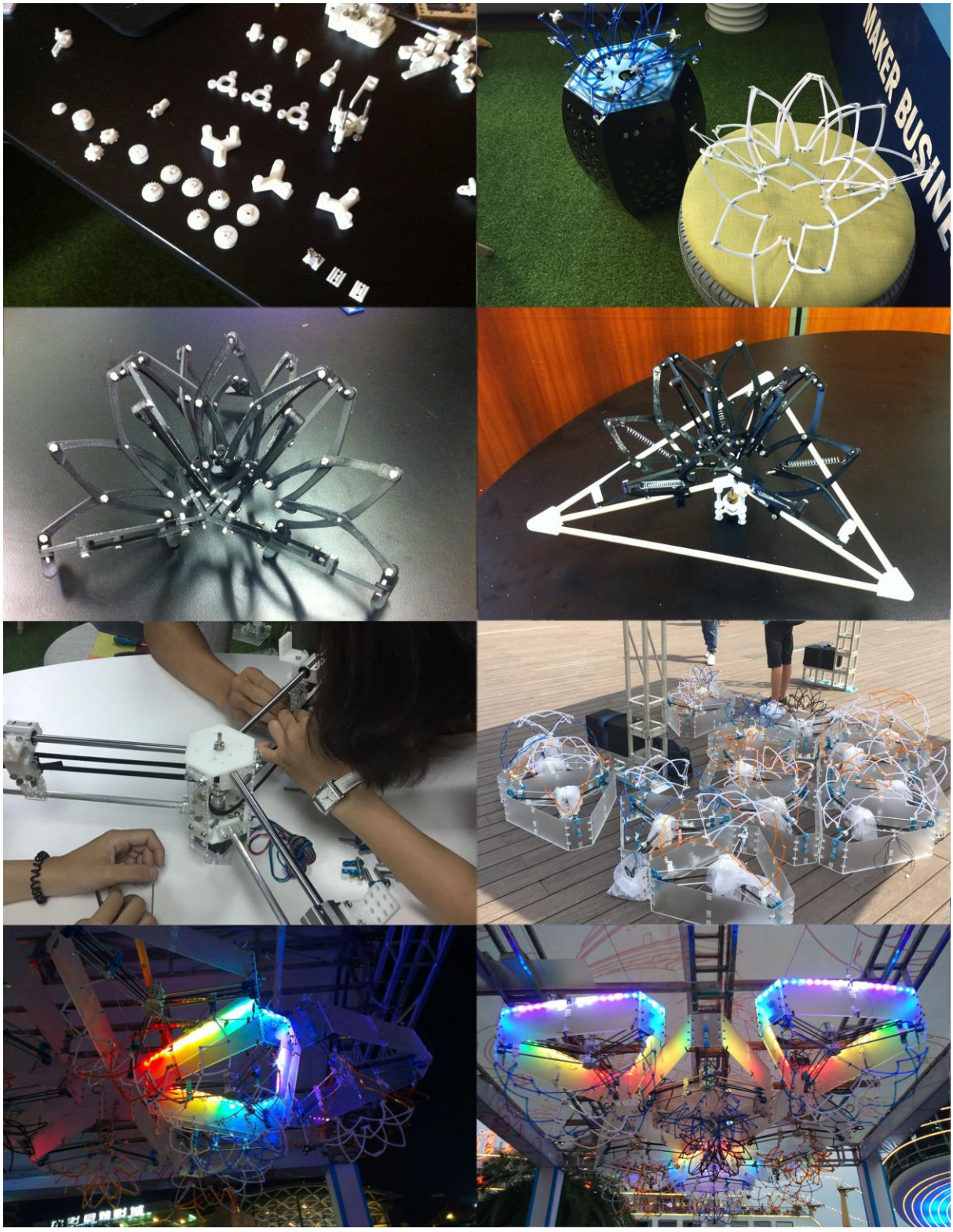






Kinetic Form- Assembly footage from Taiwan and Shenzen

Kinetic Form- Original assembly tests in Perth (top left &middle-top) & on-site testing and construction (bottom and middle-bottom).



E. General research timeline