Sequencing in Transformation of Rasterized 3D Models

Submitted by

Wei PAN

Thesis Advisor

Dr. Lujie CHEN

Engineering Product Development

A thesis submitted to the Singapore University of Technology and Design in fulfillment of the requirement for the degree of Doctor of Philosophy of Engineering, Engineering Product Development

March 2, 2018

Declaration of Authorship

I, Wei PAN, declare that this thesis titled, "Sequencing in Transformation of Rasterized 3D Models" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Abstract

Engineering Product Development

Doctor of Philosophy of Engineering

Sequencing in Transformation of Rasterized 3D Models

by Wei PAN

This thesis studies the sequencing problem of shape transformation of rasterized 3D structures. Compared with shaping (shape generating), shape transformation is a more general idea including construction, material reuse etc. To evaluate the feasibility of automatically transform by machines, this thesis investigated two operations and three cases studies: the problem of pick and place operation for homogeneous components (voxels), heterogeneous components (bricks) and folding operation for voxels. Within each case, strategies are designed to transform these problems into some mathematical models, some of which turned to be NP-complete problems such as assignment problem, traveling salesman problem etc. Mathematical principles are applied to these math problems to get optimized solutions which are incorporated in the sequencing. For instance, Hungarian method for assignment problems and Hamiltonian path searching skills for traveling salesman problem.

The optimized sequences generated by each case can be applied to achieve shape transformation either manually (nowadays) or automatically (in the future). We believe in the future the advanced building materials and tools will be developed in construction and manufacturing industry, and these sequencing and automation strategies can be directly put into applications for autonomous construction and reconstruction work.

Preface

The automatic shape changing of voxelized 3D structures is always a fancy idea for me. I have been dreaming of owning elements like what appeared in movies "Big Hero 6" or "Transformers: Age of Extinction". The emerging game Minecraft proved the popularity of this kind of voxelized style constructions. Therefore, I focused my research on these rasterized 3D models. Actually, problems related to rasterized 3D models are not easy because they are usually discrete problems.

Discretion is the one of the most important features of rasterized shape. The opposite of discrete is continuous. Compared with discrete mathematics, continuous mathematics is classical, well established, with a rich variety of applications. Discrete mathematics grew out of puzzles and then is often identified as NP-complete or NP-hard problems in some specific application areas like computer science or operations research. Basic structures and methods of both sides of our science are quite different. With the wider application of computer in various scenarios in our daily life, the solution for discrete problems are more and more important.

Fictions probably will remain factious but it may guide and inspire the development of science and technology. There is a group of studies in robotics called self-reconfigurable modular robots, which consists of many modular robots aiming to achieve the idea in the above paragraph. They are autonomous kinematic machines with variable morphology, and able to deliberately change their own shape by rearranging the connectivity of their parts, in order to adapt to new circumstances. Due to the restriction on the physical environment, initially, my research was on the locomotion and shape-changing algorithm of these physical modules. There are plenty of studies and prototypes of such robots, with different functions using for various scenarios. As I looked into these studies, I discovered that these system varies a lot due to the restricted space on hardware design, thus it's really impracticable to implement a general algorithm which suits all these prototypes. Therefore, I turned the research ideas to centralized controlling systems, and adapted to some more practical applications.

The transformed idea becomes a construction problem, and I focus on the optimization work on several metrics like material reuse rate and cost. However, there was a growing frustration that the voxelized 3D structures are always subject to physical constraints such as gravity. Unlike in the game Minecraft building cubes can be overhanging, in reality, they cannot. Therefore, my first work on voxels is applicable to 3D shapes with any overhang structures, while most 3D structures do. So I merged the voxels to bricks and developed the automatic shape changing algorithms on bricks, and it relates to the study on LEGO bricks. The folding work is inspired by a lamp design in an exhibition, and its practicability is demonstrated by a toy called rubik snake.

The results achieved and displayed in this thesis are applicable to automation industry. I have a strong belief that uniform reusable smart building materials will replace steel concrete and bricks, becoming a trend in the future.

Acknowledgements

I would like to thank my Ph.D. supervisor, Dr. Lujie Chen, for his generous guidance and support throughout all the years of this research topic. He offered me with tremendous freedom to work on problems that intrigued me. The way he conducts research, like how to analyze, “divide and conquer", and insightfully explain a problem has significant impact on me. Our group is not big (actually only two of us), which enables me to have a complete and rigorous research experience. With his supportive but quiet, welcoming while patient personality, my PhD is a invaluable and unique experience.

I also want to extend my gratitude to Singapore Ministry of Education as well as this young University, SUTD, who provided with me munificent economic support and excellent environment for this research. The small but all-around beautiful campus is one of the most important parts of my PhD journey. Of course, it includes the monotonous food provided by our unforgettably impressive canteens.

I appreciate every person who was accompanying with me and involved in my life during these years. Liyana, thank you for your thoughtful support and help in the early days I came to Singapore. You are the first sunshine shed into my PhD life here. Yan chao, thank you for your patiently discussing problems and sharing ideas with me during my research. I would like to thank Song Sibo, Dong Fei, Zhou Yiren, Mei Sen, Wang Jingyi, Li Jiaying, Guo Chu, Howard Yang, Liang Shijun, Jitendra, Kostya and Ihor, David Anderson, Ozge, Joshua Kettlewell, He Yuejun, Li Xiaoqian, Zhou Yuren and Liu Rui, James, Yoke, Max, Liu Zuozhu, Shi Yilei, Ren Zhihua, Zhou Yinning, Luo Rongrong, etc. for all of you made my PhD a complete and pleasant journey.

Special Thanks give to those who involved in my sporty life. Thank you Aloysius for organizing soccer games for us every sunday afternoon. Thanks to Jiaying, Özgun, John Caste, Surya, Xiao, Huang Wei, Ban Liang brothers, and all of those who teamed with me for soccer games. Thank you Benjamin Petry, Liu Junhua, Samuel Ng, Triss, Wen qi, Siti, Shuai Shuai and Professor Zhang Yue for playing badminton with me on Wednesday or Friday nights. As for me, these sports moments provided with me not only the joy and the fun, the relaxed time and the healthy lifestyle, but also the skills to

accomplish teamwork and the spirits for competition. Most importantly, they bring a lot of passions for my PhD life.

I am thankful to Shu En, who brought me to the COGS and our Holland cell group. It is like a family for me in Singapore. I felt blessed to have all of you around when I was in desperation. Thanks for every single prayer for me. Thank you Ps. Joseph and auntie Dawn, auntie Winnie, uncle Wee Seng and aunty Cheng Hiang, uncle Simon and auntie Patricia, Jonathan and Joanna, Paul and Vera, Kenneth and Shaun, Bryan and Mellisa, Martin and Rachel, Yi Fan, Cassandra, Joycelyn, Ernst, etc. You are always my family members in my eyes. Samual Fang, you never failed to come to me with alcohol when I felt frustrated. Benjamin Petry, you are not only my PhD schoolmate, but also my cell-mate. Thank you for your scrutinized comments for my papers and thesis, your earnest advices and your friendship.

Finally, thanks to my wife, Cynthia Xing, who has been bearing every PhD grinding moment of mine and offering hug for me whenever I am in need. Thanks to my parents who have been an ongoing source of encouragement and support. My gratitudes come from the bottom of my heart, for their constant and unconditional love. This PhD journey was not possible without any of you.

Contents

List of Figures

List of Tables

List of Abbreviations

AiC	**A**utomation **i**n **C**onstruction
AiM	**A**utomation **i**n **M**anufacturing
AM	**A**dditive **M**anufacturing
CA	**C**ellular **A**utomata
CAD	**C**omputer **A**ided **D**esign
CAM	**C**onstruction **A**nd **M**anufacturing
	Computer **A**ided **M**anufacturing
CNC	**C**omputer **N**umerical **C**ontrol
CIC	**C**omputer **I**ntegrated **C**onstruction
CLA	**C**heapeast **L**ink **A**lgorithm
DoF	**D**egree **o**f **F**reedom
FT	**F**lexible **T**ransform
HP	**H**amiltonian **P**ath
HC	**H**amiltonian **C**ircuit
GA	**G**enetic **A**lgorithms
GOS	**G**lobal **O**ptimization **S**trategy
GSS	**G**reedy **S**election **S**trategy
LED	**L**ight **E**mitting **D**iodes
LoA	**L**evel **o**f **A**utomation
LOS	**L**ocal **O**ptimization **S**trategy
MRS	**M**odular **R**obotic **S**ystems
MSR	**M**odular **S**elf-reconfigurable **R**obots
NNA	**N**eareast **N**eighbor **A**lgorithm
PCB	**P**rinted **C**ircuit **B**oard
PnP	**P**ick a**n**d **P**lace

RNNA	**R**epetitive **N**earest **N**eighbor **A**lgorithm
RP	**R**apid **P**rototyping
RSS	**R**andom **S**election **S**trategy
RT	**R**igit **T**ransform
SA	**S**imulated **A**nnealing
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
SMD	**S**urface **M**ount **D**evice
SMT	**S**urface **M**ount **T**echnology
STL	**ST**ereo **L**ithography
TSP	**T**raveling **S**alesman **P**roblem

For my family, especially my wife Cynthia Xing

Chapter 1

Introduction and Motivation

"Do not fear to be eccentric in opinion, for every opinion now accepted was once eccentric."

— Bertrand Russell

1.1 Automation in Construction and Manufacturing (CAM)

Both construction and manufacturing are essential in industries. They are similar in the way that both involve manual labor or machines to product something for commercial purposes. However, construction is usually differentiated from manufacturing, in which a product is designed for mass production; construction products are instead large and unique in form (Saidi, Bock, and Georgoulas, 2016). Construction generally refers to the creation of physical structures such as buildings, bridges or roadways. Manufacturing typically refers to the production of finished goods sold to distributors, retailers or consumers. Compared with automation in construction (AiC), automation in manufacturing (AiM) is easier to cope with thus better developed. This is because the manufacturing site in most cases are much smaller, the tolerance of structure is bigger compared with architectures in construction industry. Moreover, it can do without the messy on-site construction scene. To make things clear, the difference of them are ignored in this thesis.

As one of the oldest industries, construction industry, is facing a low productivity problem compared to other industries (Fulford and Standing, 2014). Over years, the development and advances of construction industry is slow, and it's evident that nowadays level of AiC is very low in comparison with the exiting technological advances (Balaguer, 2000), and Robotics in manufacturing industry is an evolution while

the robotics in construction industry is the not yet finished revolution. On the contrary, a high accident rate on-site results from the low labour efficiency and insufficient work quality and control in both construction and manufacturing industry. Figure 1.1 shows the fatal and non-fatal accidents rate at work in European Union. Construction is the leading sector in terms of fatal accidents while manufacturing is the leading sector in terms of non-fatal accidents. All in all, the construction and manufacturing industry can be further or fully automated to enhance its efficiency and reduce the number of accidents.

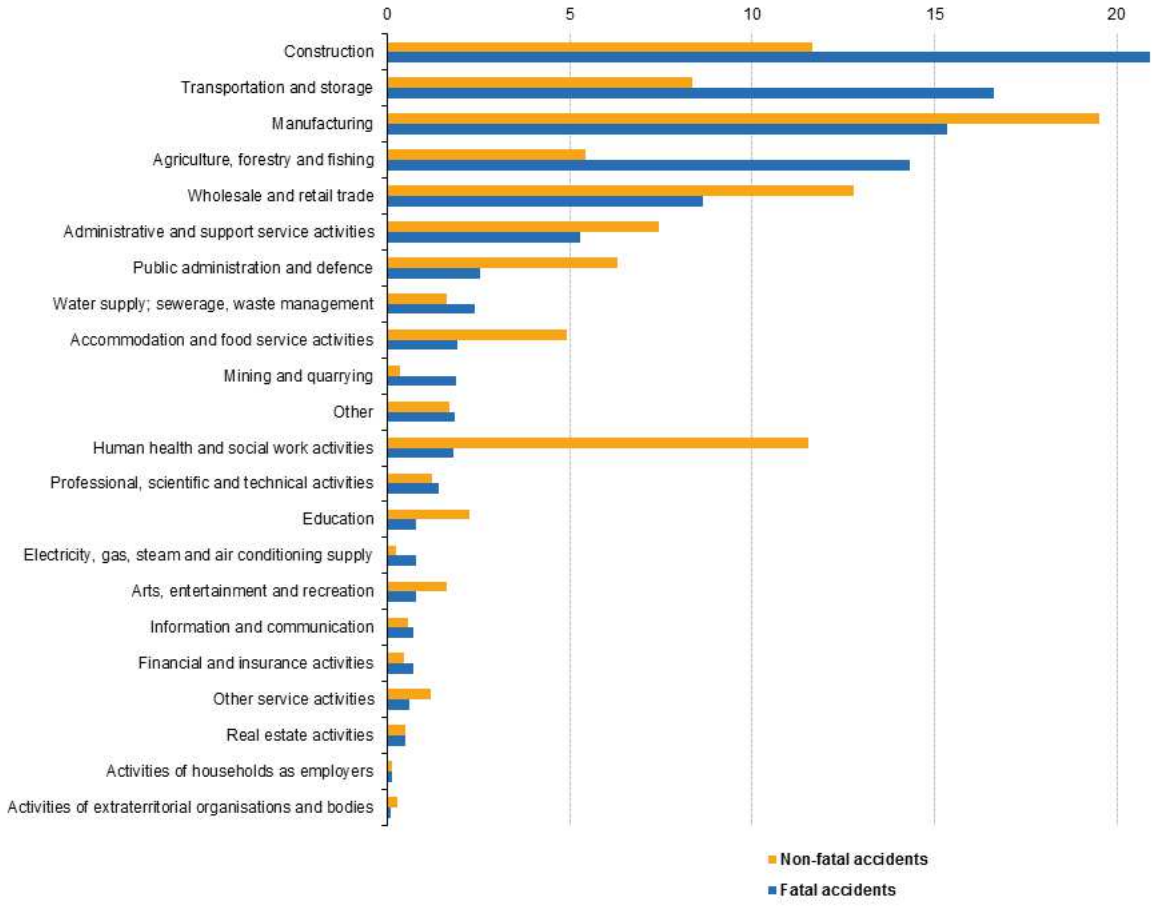

FIGURE 1.1: Fatal and non-fatal accidents at work by economic activity in European Union. Source: (*Accidents at work statistics* 2014)

Robots are widely accepted as a potential solution to improve the construction industry efficiency and reduce the number of fatal accidents of masonry works (Castro-Lacouture, 2009). Robots have technical features to enhance quality and efficiency of

the operations, and potentially perform construction tasks where human presence is impossible, undesirable, or unsafe. For instance, construction in hazardous areas such as ground after earthquakes and nuclear accidents, construction under difficult physical conditions such as undersea or outer space. While modern robots were born in 1940's, and becoming dominant in mass production lines, it's hard to find any evidence of their application in construction industry until late 1980's (Hagis, 2003; Ichbiah, 2005). Although there are plenty of documents related to the history of the application of robots (Bidgoli, 2015; Balaguer, 2000) in construction, a brief introduction is given as following.

Phase 0: Ancient Approach (Before the 1970s) The construction of the middle age cathedrals in Europe was done by simple construction technology (Fig 1.2). The bricks and columns elaboration were performed manually and on-site. Their elevation was also performed manually using simple mechanisms, like fixed pulleys and ropes (Follett, 2010). The material supply, the transportation and the assembly technology were very important and challenging, which result in a long construction period (even centuries).

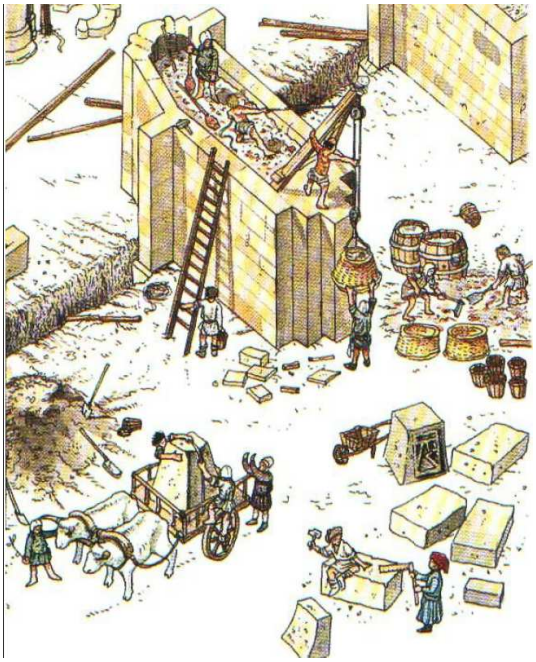

FIGURE 1.2: Middle age cathedral construction. (Follett, 2010)

Phase I: Construction Automation Approach (the 1980s and 1990s) By the end of 1980's, several Japanese construction companies, including Fujita Corp., Obayashi Corp., Kajima Corp., etc. invested on using custom-made robots for on-site constructions like picking and placing concrete slabs and painting façades. For instance, Kangari (Kangari and Yoshida, 1990) introduces several prototypes of construction robots developed by Shimizu Corp. to reduce the manual work and optimize the building process, as shown in Figure 1.3. Compared with nowadays constructive machines, their machines were highly specialized, extremely expensive, and suffered from the lack of flexibility.

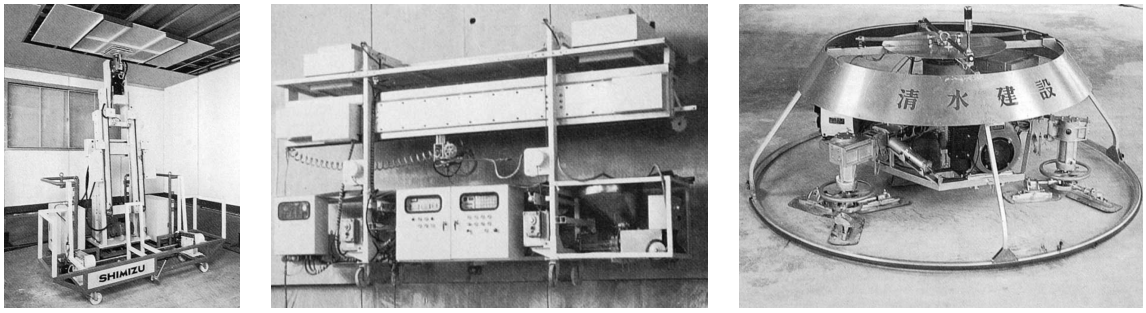

FIGURE 1.3: Prototypes of construction robots developed by Shimizu Corp. (Kangari and Yoshida, 1990) (a) Ceiling panel positioning robot (CFR-1). (b) Exterior wall painting robot. (c) Concrete plastering robot.

Phase II: Construction Automation Integrated with Computational Machinery (the 2000s) With the emergence and development of computers, more complicated system and advanced controlling tools are developed. The rapid integration of computer numerical control (CNC) machines significantly steered the direction of robotics exploration in architecture and construction, which results in a modern construction concept: Computer Integrated Construction (CIC) (Miyatake and Kangari, 1993). Furthermore, construction sites were better structured and designed like factories, and the on-site construction was implemented as an semi-automated assembly line (Tezuka and Takada, 1992). Strong complementaries exist between the actual building – its design, manufacturing and information technology – and its construction and organization strategy. The performance and scale of construction has been significantly enhanced (Moselhi, 1998; Linner and Bock, 2012; Linner, 2013).

Figure 1.4 shows an on-site masonry robot (a) and the partly automated construction site of Tokyo Sky tree.

FIGURE 1.4: On-site construction robots and automated construction site. (a) Thomas Bock, ESPRIT 3 6450 ROCCO (RObotic Computer integrated COnstruction) masonry robot, Karlsruhe University, Germany, 1992–6: Development of a mobile heavy-duty robot for the construction sector, here applied to robotic on-site assembly Bock and Langenberg, 2014. (b) Nikken Sekkei, Tokyo Sky Tree, partly automated construction site, Tokyo, 2010: The 32-storey, 634-metre (2,080-foot) tall Tokyo Sky Tree radio and television tower in the city centre was built and is operated by Obayashi Corp.. (Bock and Langenberg, 2014)

Phase III: More Advanced Construction System (from now on) Interdisciplinary study is highlighted in nowadays science and technology. Swarm intelligence (Wawerla, Sukhatme, and Mataric, 2002) and drones (Augugliaro et al., 2014) are making differences in automated construction. In some recent studies (Willmann et al., 2012; Mirjan et al., 2013; Lupashin et al., 2014; Fink et al., 2011), unmanned aerial vehicles (UAVs) carried bricks from palettes to designated positions to build structures based on predetermined sequences. In another study (Werfel, Petersen, and Nagpal, 2011; Werfel, Petersen, and Nagpal, 2014), a swarm of robots assembled structures by mimicking the behavior of social animals such as termites. For industry usage, Amazon

company applied a system in 2012 in which goods are stored on portable storage units, and carried by swarm robots (D'andrea et al., 2010).

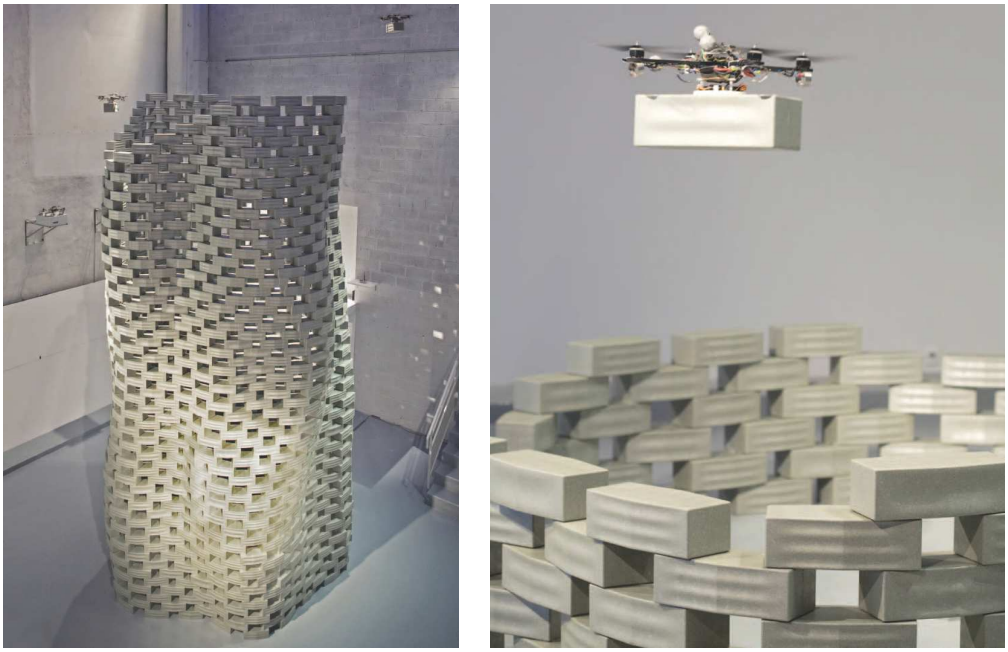

FIGURE 1.5: The Flight Assembly System (Augugliaro et al., 2014). (a) The Flight Assembled architecture installation. The 6-m-tall tower consisting of 1500 foam elements was assembled by four quadrocopters in France, 2011. (Photograph by François Lauginie.) (b) Placing a module. The vehicle hovers above the structure before placing a construction module.

Most of these advanced construction systems are not capable for currently construction industry, restricted to the environment, size, specialized building blocks, etc. However, in light of novelty and intelligence, they can confidently steer the direction and trend for the automation construction and manufacture in the future.

1.2 Material Reuse and Sequencing Optimization

Material reuse Another initiative of this thesis is the design of material reuse for construction and manufacturing. As it is known to us, the abuse of resources such as plastics, construction materials, etc. has caused problems like pollution to human society. It takes a large sum of funding and efforts to finish deconstruction and depose of

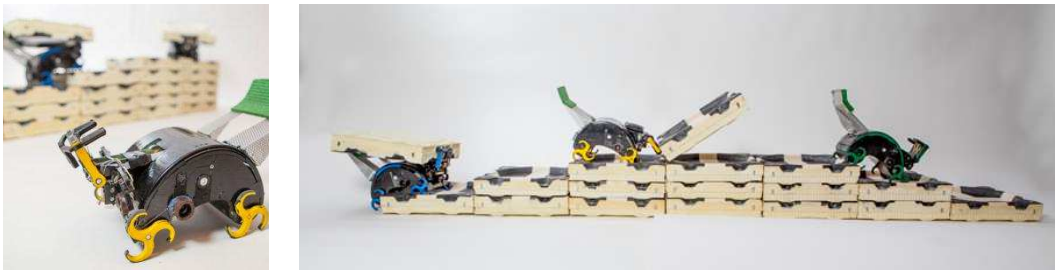

FIGURE 1.6: Self-organizing robots inspired by termite colonies demonstrate swarm-like intelligence (Werfel, Petersen, and Nagpal, 2014) (a) The robots can build themselves staircases to reach the next construction points, and they know how to add bricks that advance construction without blocking important paths. (b) The TERMES robots can carry bricks, build staircases, and climb them to add bricks to a structure, following low-level rules to independently complete a construction project. (Photo by Eliza Grinnell, SEAS Communications.)

the waste. The potential use of various solid wastes for producing construction materials (Safiuddin et al., 2010; Tam and Tam, 2006) and the recycling of these construction materials (Roper, 2006) have been investigated. However, so far there is no research conducted on the reuse of these construction materials such as bricks and reinforced bars. In the meanwhile, the topic of material reuse keeps arousing good inspiration of arts and designs (Addis, 2012; *Canstruction*), as shown in Figure 1.7.

It can be anticipated that replacing the current building blocks, cement, and bricks by super-bricks connected through new glue (similar to Lego bricks) would significantly ease the construction and deconstruction processes, and as a consequence, brick-reuse will be an important issue.

Sequencing Optimization In computer science, sequencing refers to sequential processing of a set of commands. To avoid ambiguity, in this thesis, sequencing refers to the process that by following a specific set of rules, to generate a sequence that accomplishing the construction task, i.e. to build a sequence of picking and placing a set of cubes.

While humans rely on intuition to create a sequence to assemble structures from parts, this process can be augmented through intelligent devices (Anderson et al., 2000; Gupta et al., 2012), simple assembly of parts may be achieved by a set of actuators without human intervention (Schoessler et al., 2015).

FIGURE 1.7: Examples of arts and architectures made of reused materials: (a) A pavilion constructed in the campus of the University of Applied Sciences in Detmold. It consists of more than 2000 beer boxes. (b) The Luxury Pavilion Built From Recycled Bedsprings, displayed in Abwab 2017, the highlight of Dubai Design Week that exhibits designers' talent from across the Middle East, North Africa and South Asia. (c) Front-page photo of Canstruction (*Canstruction*). "An exhibition of structures made from unopened cans of food that are later donated" - New York Times

The work of Dierichs (Dierichs and Menges, 2016; Dierichs and Menges, 2017) illustrates the importance of moving sequences (Figure 1.8). The plastic granular materials are picked and placed in a sequence while they grip and bind to one another when stacked. An optimized PnP sequence will benefit the autonomous system in light of the efficiency and robustness of the structure.

Construction done by multiple robots in parallel raises challenges in efficient motion sequencing (Werfel, Petersen, and Nagpal, 2011; Lindsey, Mellinger, and Kumar, 2011). Algorithms exist to minimize the workload imbalance between the robots and to maximize assembly parallelization (Worcester, Rogoff, and Hsieh, 2011).

FIGURE 1.8: The ICD Aggregate Pavilion 2015 from the University of Stuttgart. They used 30,000 spiky components and a robot to create a pavilion described as the "first architectural structure realized with a designed granular system".

1.3 Rasterized 3D Shape

In computer graphics, just as pictures are consist of pixels, 3D shapes can be represented by voxels (Jan, 2016). Normally 3D models are represented by polygonal meshes or points cloud, it can be further converted into voxels representations by rasterzing (voxelizing) techniques (Cohen-Or and Kaufman, 1997; Kaufman, Cohen, and Yagel, 1993), as shown in Figure 1.9. Rasterized 3D data are also termed as volumetric data (Sramek, 1996), which are widely applied in Gaming industry, Geographic Information System(GIS), Arts and Designs etc.

In fabrications and constructions, rasterized 3D representations are useful in the same way as buildings may be constructed from bricks. This is similar to a popular computer game, Minecraft. In this game, building blocks are collected and piled by players to build all kinds of architectures and arts (Duncan, 2011). Online instructions are given to build these buildings or structures in games. These instructions are provided as blue prints layer by layer in a sequence, which is similar to LEGO bricks (Figure 1.10). In this case, the virtual figures controlled by the player is committing the construction work.

It has been shown that rasterized structures can be built by automated machines,

which have already played important roles in the advanced manufacturing industry; however, in the construction industry automation is primitive due to an economical reason (Balaguer, 2000; Helm et al., 2012): it is often more expensive to use machines, e.g. robots, than men. In this thesis, rasterized 3D structures are investigated for construction sequencing for the following reasons.

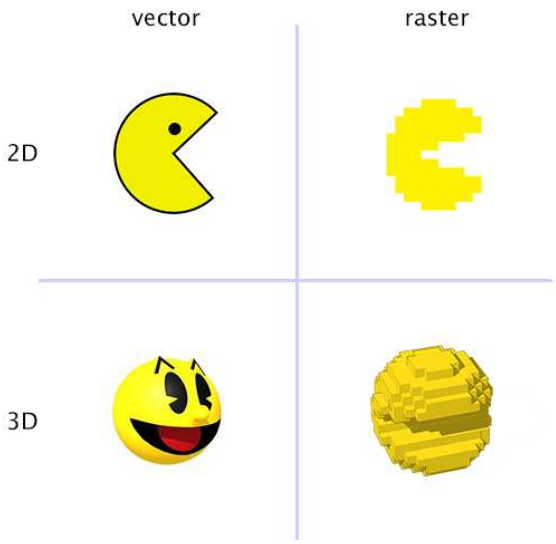

FIGURE 1.9: 4 quadrants to distinguish 2D/3D space and vector/raster representations in computer graphics (Jan, 2016).

Consistency with physical components Voxels are not only easy to process as computer data, but they can also be instantiated by physical components for construction, such as cubes, bricks, food cans, etc. Varied with different resolutions, any 3D shape can only be realized in the form of its discretized representations, a.k.a., voxels.

Reusability Apparently voxels and bricks can be easily reused, compared with heterogeneous building materials. A good example is LEGO, a globally popular toy composed of colorful plastic bricks that can be interlocked and assembled in many ways. (Figure 1.11)

Mass Production Homogeneous components such as bricks enable mass production, which tremendously reduces the cost of products from the end product (in this case, the bricks) as well as from the manufacturing equipment (robots, machine tools, etc.).

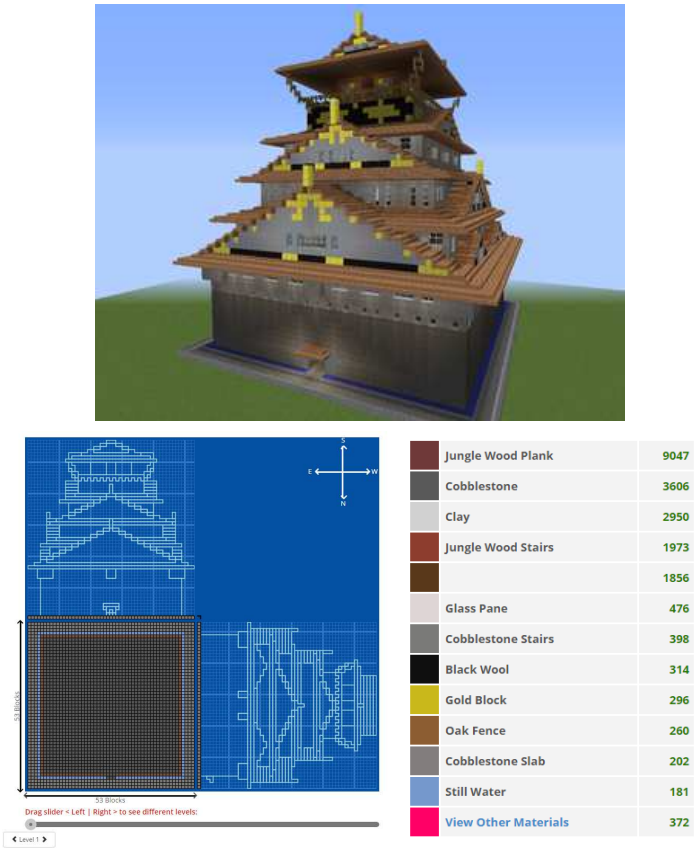

FIGURE 1.10: Online instruction to build Feudal Japanese Osaka Castle with 21931 blocks. (Zal, 2017) (a) Feudal Japanese Osaka Castle, (b) Blueprints (bottom layer) of the building instructions, and (c) Applied material blocks.

FIGURE 1.11: (a) a box of basic bricks from an official LEGO shop. (b) A mess of unsorted Lego bricks.

1.4 Building Components

There are different kinds of materials for construction and manufacturing. They can be categorized as following.

1.4.1 Homogeneous & Heterogeneous

The building components can be homogeneous or heterogeneous. For instance, bricks used in building constructions are homogeneous, which are easy for mass production and reusing. However, the disadvantage is the restriction on the special functionalities in some specific scenarios such as the roof of the building (tiles).

Voxel Voxels are homogeneous building components. This term describes an abstract concept which encompasses cubes (Figure 1.7a), bricks (Figure 1.5), hexagonal pillars, cans (Figure 1.7b), octet truss (Deshpande, Fleck, and Ashby, 2001), etc. Voxels are generally applied as building materials in pyramidal shape. A shape is pyramidal if it has a flat base with the remaining boundary forming a height function over the base (Hu et al., 2014). Pyramidal shapes are optimal for scenarios like molding, casting, and layered 3D printing, which are thereby good for constructions or fabrication by voxel components. For example, a normal approach called 3D voxel printing is proposed by Hiller et. al. (Hiller and Lipson, 2009).

In some cases, with mere connections with neighbors from either upper layer or below layer, the constructed 3D Shapes are usually unstable.

Lego Brick As a type of popular edutainment toy, LEGO bricks can be regarded as heterogeneous building components (Figure1.11). Its application has been found in research ideas relating to software engineering (Feijs and De Jong, 1998), rapid prototyping (Mueller et al., 2014), material science (Celli and Gonella, 2015), and nucleic acid nano-engineering (Ke et al., 2012; Gothelf, 2012). LEGO bricks are inherently suitable for constructing 3D shapes: they enable diverse part combinations; they are in various colors; and they can be reused; however, the complexity of LEGO construction increases significantly with the size of a 3D shape. Although many ways of part combination are feasible, it is difficult to build an accurate LEGO model without step-by-step instructions.

A LEGO construction problem was proposed in 1998 (Gower, Heydtmann, and Petersen, 1998): Given any 3D body, how can it be built from LEGO bricks? Nowadays, the problem is largely solved owing to several computational algorithms (Kim,

Kang, and Lee, 2014; Petrovic, 2001; Smal, 2008; Ono, Alexis, and Chang, 2013; Testuz, Schwartzburg, and Pauly, 2013). For automotive LEGO construction, the stability of the intermediate and constructed LEGO structure and feasibility of the fabrication are both very important. Some researches have been done to solve the stability (Luo et al., 2015; Hong et al., 2016) and feasibility (Zhang et al., 2016) problems. Moreover, Pasek and Yip-Hoi (Pasek and Yip-Hoi, 2005) proposed an interesting computer integrated manufacturing system based on LEGO bricks.

LEGO is just a toy, but the concept of an automotive bricks construction system can be inspired from it. In the future, there might be more advanced LEGO-like building components which requires optimization strategies to minimize the steps of building and transforming by reusing bricks and to maximize the reuse rate bricks.

1.4.2 Lattice & Chain

This classification is aroused from the grouping of crystal structure system in physics. In lattice system lattice-type components are distributed or stacked in 3D space lattices, and usually translational movements are taken effects without considering rotations.

In a chain system, components are usually connected with its two neighbors, and the shape ranging from 1D to 2D and 3D will change subject to rotational movements. Chain-type components are more common in manufacturing than construction. Figure 1.12 shows a toy invented by Ernő Rubik, called Rubik snake with twenty-four wedges that are right isosceles triangular prisms. The wedges can be twisted to resemble a wide variety of objects, animals, or geometric shapes (Fiore, 1981).

Chain systems usually can be regarded as a special type of lattice system. In some cases, components can be both lattice-type and chain-type (Figure 1.13).

1.5 Sequencing Operations

To achieve AiC, it is valuable to investigate whether there exists an optimal construction sequence that minimizes the work, and if exists, how to find it. We frame the problem as a transformation from one rasterized structure (source) to another (destination). Our objective is to find an efficient sequence that a machine can follow to

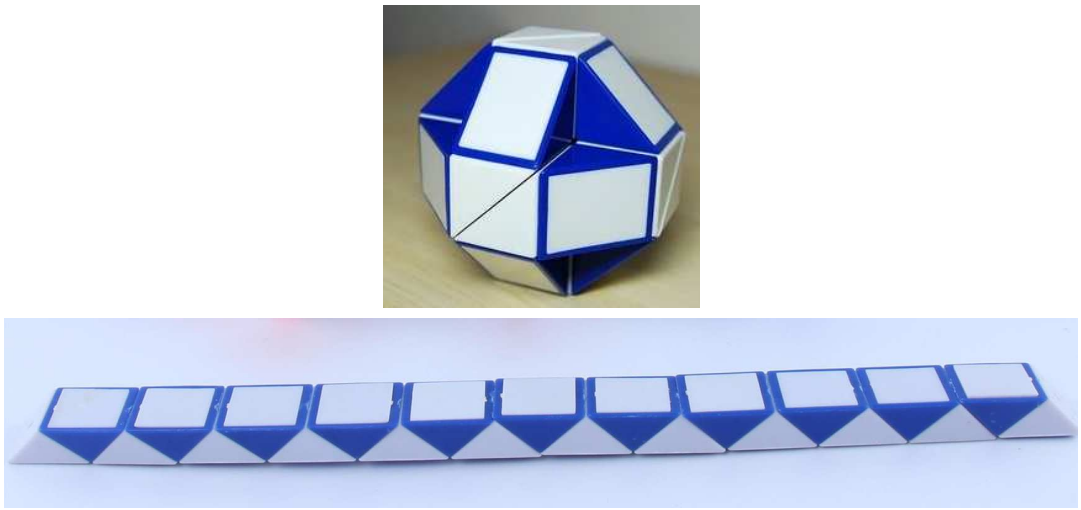

FIGURE 1.12: Rubik's Snake (Fiore, 1981). (a) Its "ball" shape in its packaging is a non-uniform concave rhombicuboctahedron. (b) Rubik's Snake as a string

FIGURE 1.13: Super Clover: an LED pendant lamp group designed by Michael Young (Young, 2017a; Young, 2017b) (a) The Clover array is a direct descendent of the Super Clover. (b) The Super Clover: a large geometric lighting form based upon extensive study of the opportunities presented by creating light sculptures within a mathematical grid system. One lighting lamp can be connected to two neighbors.

achieve the transformation, where efficiency is described by an application-dependent cost function. The framework encompasses the situation of constructing a structure from material palette; in this case, the source is the shape of the palette and the destination is the desired structure.

1.5.1 Pick and Place (PnP)

For lattice-type components, PnP usually is the best solutions to move in a sequence, which is a translational move. In the same way buildings may be constructed from bricks or pre-fabricated components, a structure can be assembled from equal-sized parts by a process called pick-and-place. Initially, PnP refers to placement of electronic components onto circuit boards (Ho and Ji, 2005). The term is broadly used in areas including architecture, industrial automation, and robotics, to refer to such processes as case packing, palletizing, machine tending, and assembly. PnP is also applied to assemble 3D structures from parts layer by layer (Glynn and Sheil, 2011; Moses et al., 2014). The resulting artifacts are of particular interest to architects and engineers who need physical prototypes at different stages and scales of design for visual and functional evaluation. The parts are often low cost, reusable, and easy to store, which supports a sustainable approach to many iterations of physical evaluation. This approach is also important in the field of 3D construction, advanced pre-fabrication and assembly. Figure 1.2, 1.4, 1.5, 1.6 are examples involving PnP process.

Bricklaying robots can pick up bricks from prepared pallets, apply mortar, and place them to correct locations (Bock et al., 1996; Pritschow et al., 1996; Yu et al., 2009). Modular-house-assembly robots can construct residential houses from 2D or 3D prefabricated modules (Leyh, 1995; Gassel, 1996; Balaguer, 2000; Diez et al., 2000), which shows the features of voxels by PnP as mentioned above. More recent work of PnP is coupling machine learning technology like object recognition with traditional robotic arms (Zeng et al., 2017). Amazon Robotic/Picking Challenge (Wurman and Romano, 2015) is a platform to relate research to industrious applications of automatic PnP process. In long term, the unmanned PnP approach may be a solution to reducing construction cost in regions where manpower becomes increasingly expensive.

1.5.2 Folding

Folding is a rotational operation, including the operation to transform strings (1D) into continues areas (2D) or volumetric shapes (3D), e.g. Rubik Snake, as well as the operation to transform continues areas into volumetric shapes with techniques, e.g. origami. Folding 1D strings into 2D and 3D shapes has been a long sought goal in a wide range

of fields in academia and industry, such as protein folding, polymer packing, cellular robotics and aspects of self-assembly and analytical geometry.

Folding is more common in cases where the construction is involved with chain-type components such as DNA and protein assemblies. A toy, called Rubik Snake (Figure 1.12), which can transform into different shapes by folding proves to be useful for understanding the operation of folding (Iguchi, 1998), with its similarities to the features of denatured protein. Paper folding, a.k.a. origami, is a popular art which transform a flat sheet of paper into a finished sculpture through folding and optionally sculpting techniques (Mitani, 2016). Theories and algorithms are well studied for Rubik Snake (Ding and Lu, 2013) and origami (Fei and Sujan, 2013; Kanade, 1980), which primarily result in the generation of sequences as an instructions.

As a sequential operation, folding can also be accomplished automatically by robots. Robots are developed that capable of folding a ribbon into 2D or 3D structures (Wang, Plecnik, and Fearing, 2016) as well as finishing origami (Balkcom and Mason, 2008). As a trend towards higher level of automation, it's meaningful to relate the study of sequencing to the robotic folding industry.

1.6 Thesis Structure

The thesis is structured as follows:
in Chapter 2 we will give an overview of related work to 3D rasterized model sequencing problems which have been realized both in construction and manufacturing domain. This will give the reader an appreciation of the current state-of-the-art of this study.
Chapter 3, 4, and 5 are three cases study in the sequencing optimization for rasterized 3D structures respectively: PnP operation on voxels representation (homogeneous), on bricks representation (heterogeneous), and folding operation on voxels. The three cases represent the majority of the sequencing problems for the transformation and shaping of rasterized structures.
Finally, in Chapter 6, we summarize this thesis with conclusion and future work.

Chapter 2

Related Work

"A journey of a thousand miles begins with a single step."

— Laozi, an ancient Chinese philosopher.

2.1 PnP Automation in Construction and Manufacturing

Firstly, some typical examples of PnP applications in industry are listed here to demonstrate the demands of automation in this field.

In the work of (Dierichs and Menges, 2016; Dierichs and Menges, 2017), granular materials made from recycled plastic are aggregated to create a vertical wall as they grip and bind to one another when stacked (Figure 2.1a). These granular components are fully reconfigurable and recyclable (Figure 2.1b). Consequently they can be reused after each construction phase, and infinitely rearranged to allow for change and adaptation during a structure's lifetime. The pick and placement processes are autonomous by applying an industrial six-axis articulated robot. An optimized PnP sequence will not only enable the aggregation of granular spiky components, but also benefit the whole system in light of its efficiency and robustness.

2.1.1 Masonry Robots

Rihani reviews some projects on robotic masonry, and presents a new initiative to study critical issues related to the establishment of a robotic mansonry system by discussing three major problem areas where computer and robots play a key role: (1) design automation, (2) automated mortar application, and (3) adaptive control of brick placement (Rihani and Bernold, 1994). Pritschow et al. presents a process for automated

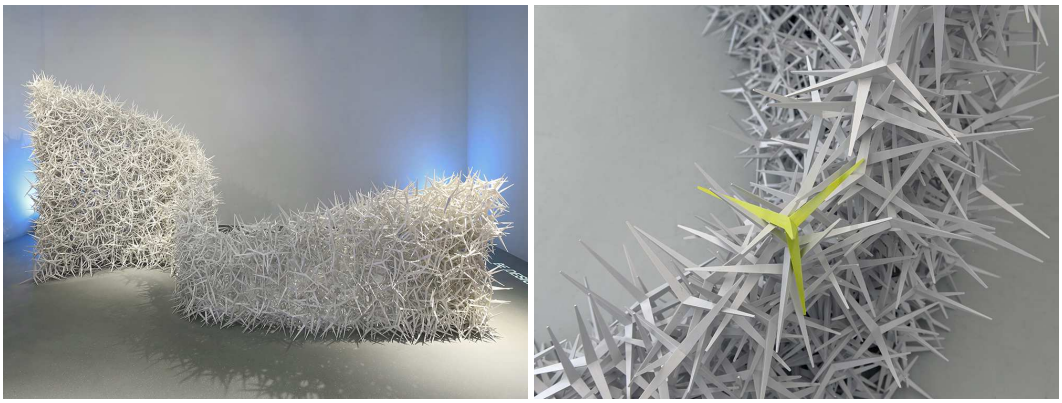

FIGURE 2.1: The ICD Aggregate Wall. (a) A vertical wall aggregated by granular components. (b) The granular components. The yellow one in the center is a special one, which is used to stablize the structure.

masonry construction on a building site by means of a mobile robot (Pritschow et al., 1996). It's actually a semi-automated system in which the on-site operation is a man-machine-system comprising the mobile bricklaying robot and a skilled worker. A wall assembly system was developed by Bock et al. (Bock et al., 1996). The construction sequence was predetermined off-line; a robot could identify the current state with a recognition module, and determine the next move of the assembly sequence. Gramazio and Kohler (Kohler, Gramazio, and Willmann, 2014) employs industrial robots for developing 1:1 prototypes that explore the potential of robotic production in architecture, the Robotic manipulator arms are used to automate the assembly of complex geometric structures.

2.1.2 Autonomous Robots

Most of autonomous construction robots are on experimental scales and they are far from the commercial stage, though mobile robots are proved to be extremely useful in the field of architecture (Willmann et al., 2012). A recent review presented by Ardiny shows the state-of-the-art research into automated construction by autonomous mobile robots (Ardiny, Witwicki, and Mondada, 2015). The relevant studies in terms of applications, materials and robotic systems are classified. Magnenat et al. (Magnenat, Philippsen, and Mondada, 2012) used the marXbot robot to grasp ferromagnetic self-alignment blocks. The odometry and laser data were used to perform simultaneous

localization and mapping (SLAM) and the front camera and proximity sensors were employed to provide the required data for dropping blocks. In the work of Wismer et al., a roofed structure was built using a VICON motion system to estimate the position of the marXbot (Wismer et al., 2012).

It was tightly interwoven with a relatively new type of construction: aerial robotic construction (Willmann et al., 2012; Fink et al., 2011), where unmanned aerial vehicles (UAVs) carried bricks from palettes to designated positions to build a tower by bricks based on predetermined sequences. They benefited from the real-time camera system to guide robots according to digital design, allowing the robot pickup and deposit objects.

It was also touched upon in research in swarm intelligence where inspirations from social animal behaviors like bees, ants, schools of fish or flocks of birds were applied to robotics (Navarro and Matía, 2012). Compared with single robot, swarm robots have advantages in parallelism, scalability, robustness and fault tolerance (Mohan and Ponnambalam, 2009), while more challenges were also introduced such as communication between agents, task allocations, etc. A typical example is done by Werfel. et al., where a swarm of robots assembled structures by mimicking social animals such as termites (Werfel, Petersen, and Nagpal, 2011; Werfel, Petersen, and Nagpal, 2014). The ground mobile robot system performs collective automated construction relying on local information and implicit coordination to align robots to a structure, and climb over obstacles to build structures using passive building blocks.

A system for transporting inventory items called Kiva is applied by Amazon (D'andrea et al., 2010). This is a set of warehouse robots each includes an inventory holder capable of storing inventory items and a mobile drive unit. The mobile drive unit is able to move to a first point and coupled with an inventory holder, like a picking operation. Then, the mobile drive unit is capable of determining a location of the inventory holder and calculate a route to place it to the destination. These studies are typical rasterized structure construction using PnP.

2.1.3 Automation in Manufacturing

Compared with AiC, AiM experienced a wide variety of development, which gives an extremely wide range on this topic. Nevertheless, there are quite a few researches reviewed on the development of AiM (Frohm et al., 2008; Kalpakjian and Schmid, 2014; Esmaeilian, Behdad, and Wang, 2016). Frohm et al. made a evaluation on the level of automation (LoA) in manufacturing (Frohm et al., 2008). Definitions and taxonomies for LoA has been investigated across multiple scientific and industrial domains. Esmaeilian et al. presented a state-of-the-art survey (Esmaeilian, Behdad, and Wang, 2016) on manufacturing systems in terms of both the tangible and intangible elements, including the influence and applications of robots on manufacturing system. They pointed out that emerging technologies such as additive manufacturing and advanced planning and scheduling are important factors in the trends to future developments of manufacturing. To be concise, only AiM involving potential sequencing optimization problems are investigated and listed as follows.

2.1.4 Surface Mount Technology (SMT)

Surface Mount Technology is a mature technology, which is applied to assemble electronic devices by soldering electronic components such as chips or resistors onto printed circuit boards (PCB) (Prasad, 2013; Strauss, 1994). As shown in Figure 2.2, the SMT assembly lines usually involve solder paste, component placement and solder re-flow operations (Tirpak, 2000), and the optimization of the assembly lines can be realized from both the surface mount devices (SMD) and the sequencial PnP operation algorithms such as feeder setup and component PnP sequences (Ayob, Cowling, and Kendall, 2002; Ayob and Kendall, 2008). Although the topic of this thesis emphasizes the latter, the two factors are usually inter-knitted that a specific sequential algorithm is usually designed for a specific SMD placement machines or an optimized sequential idea arouses the setup of SMD.

The first PnP SMD machine was introduced in the early 1980s, and since then it

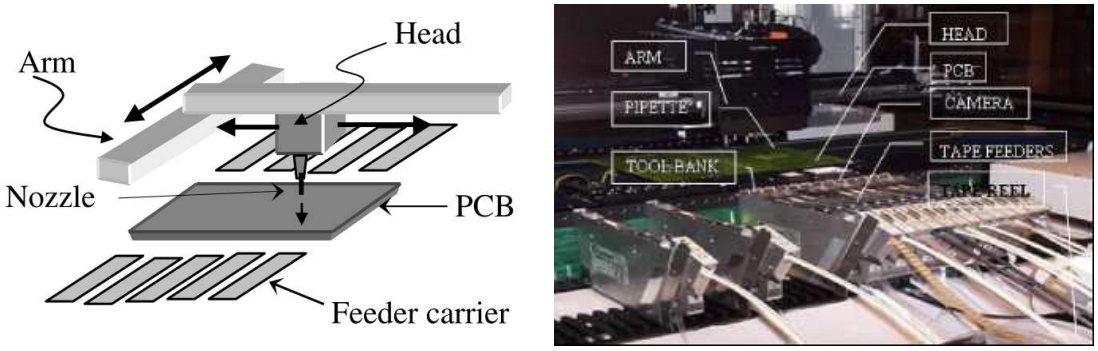

FIGURE 2.2: Examples of SMD PnP machines (Ayob and Kendall, 2008). (a) A sequential pick-and-place SMD placement machine. (b) An example of an SMD placement machine (Dima HP-10).

was keeping involving even nowadays due to significant demands on electronic devices (Strauss, 1994; Bentzen, 2000). Up to now, there are various SMD placement machines available, such as sequential pick-and-place, rotary disk turret, concurrent pick-and-place, etc. (Grotzinger, 1992; Khoo and Loh, 2000). In general, the component PnP sequencing problem is modeled as a traveling sales-man problem (TSP), which is an NP-hard problem (Ayob and Kendall, 2008; Truss, 1998). By incorporated with a TSP algorithm, De Souza et al. proposed a knowledge-based component placement system to solve the PnP sequencing problem for a SMD placement machine (DE SOUZA and LIJUN, 1994; De Souza and Lijun, 1995). By grouping the components by type, a quantity threshold and the device size consecutively, their algorithm obtained a 24% improvement of the board travel distance.

2.2 Modular Self-reconfigurable Robots (MSR)

PnP motion planning has another related area in robotics: modular self-reconfigurable robots (MSR) or Modular Robotic Systems (MRS) (Murata and Kurokawa, 2007; Yim et al., 2007; Feczko et al., 2015). A MSR system often consists of many equal-sized modular robots, which can be reconfigured into different shapes of a robot to achieve various functions. Each module may have actuators, sensors, processors and ways to communicate with its neighbors so that shape transformation is realized autonomously. During transformation, the motion of each module is generated with certain strategy.

Because the module of MSR is expensive (compared with other building components, e.g. bricks), it's usually employed as components for assembling rather than construction work. However, the algorithms applied by MSR may involve the construction sequencing problems, because of the modular feathers: (1) rasterized feature; (2) modules/components reuse; (3) collective behavior; (4) metrics in the evaluation of algorithm. The development of MSR may provide inspiration and guidance for material science in direction of new construction materials. Therefore, it's meaningful to put an eye on this new area.

The planning algorithm of MSR are reviewed in the work of Ahmadzadeh et al. (Ahmadzadeh and Masehian, 2015). They categorized the self-reconfiguration planning algorithm into four classes: search-based, control-based, agent-based and bio-ispired. Readers may go through their work for further information. As for the MSR systems, Yim et al. investigated the state-of-the-art of MSR systems and categorized them into two types: lattice type and chain type (Yim et al., 2009).

2.2.1 Lattice Type

Gilpin et al. designed MICHE modules (Figure 2.3a). The algorithm for the Self-disassembly is based on local communications between modules through five steps: (i) Neighbor discovery: modules are added to an initial assembly in an order, and upon insertion, each module immediately commences to find its neighboring modules and establishes magnetic connections to them ; (ii) Localization: each module locates itself in the assembly and sends its location data, i.e. its coordinates to a host computer where a 3D model of the initial configuration is displayed; (iii) Sculpting: according to the desired shape, an operator virtually sculpts the block at the host computer, and sends back the shape information to the system; (iv) Shape distribution: a message is sent to a single root module which controls other modules in the system; and (v) Disassembly: any module that is located off in the last step disconnects itself from the system (Gilpin et al., 2008). Later, an enhanced version was designed as Smart Pebbles modules (Figure 2.3b), without the restriction of the unique root module during self-disassembly process. (Gilpin, Knaian, and Rus, 2010; Gilpin, Koyanagi, and Rus, 2011).

In addition, it is able to produce multiple shapes, with no restriction in the number of modules.

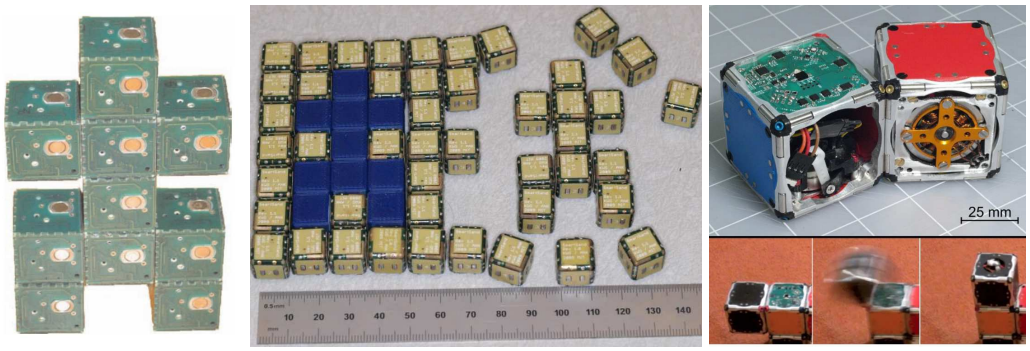

FIGURE 2.3: (a) A robot-like shape self-disassembled with 10 Miche modules. (Gilpin et al., 2008). (b) Robot Pebbles (Gilpin, Knaian, and Rus, 2010). (c) M-Blocks (Romanishin, Gilpin, and Rus, 2013).

Romanishin et al. designed M-Blocks (Figure 2.3c), a novel self-assembling, self-reconfiguring cubic robot that uses pivoting motions to achieve locomotion. By quickly transferring angular momentum accumulated in a self-contained flywheel to the body of the robot, each individual module can pivot to move linearly on a substrate of its neighboring modules and move independently to traverse planar unstructured environments (Romanishin, Gilpin, and Rus, 2013).

2.2.2 Chain Type

Yim developed the Polypod robot system (Yim, 1994) which consists of two types of modules: segment and node. A segment has two degrees of freedom (DoF): two connection plates at opposite ends while the node modules have no DoF, but six connection plates and contain a battery (Figure 2.4a). Castano et al. Developed the CONRO (CONfigurable RObot) module (Castano, Chokkalingham, and Will, 2000), which has a female connector at one end and three male connectors located at the other (Figure 2.4b). Each module has two DoF: pitch (up and down) and yaw (side to side), which is controlled using hormone-inspired distributed controllers.

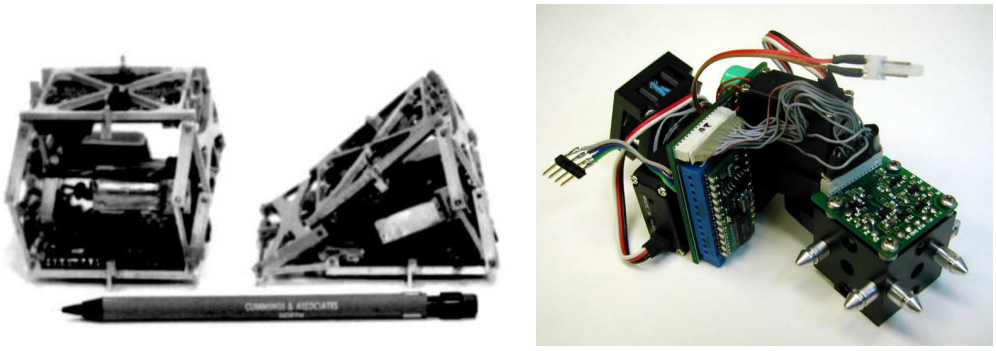

FIGURE 2.4: (a) A Polypod segment and node module (Yim, 1994). (b) A close-up of a CONRO module (Castano, Chokkalingham, and Will, 2000).

2.2.3 Hybrid

The MTRAN system is a highly maneuverable and reconfigurable system developed by Murata et al. (Murata et al., 2002), which combines the positive capabilities of chain and lattice based systems (Fig 2.5a). Each module has two kinds of boxes: active and passive. The active cube can pivot about the link that connects them and can form chains for performing tasks. However, during reconfiguration, each of a module's two cubes can occupy a discrete set of positions in lattice space when attempting to align with another module and bond for reconfiguration.

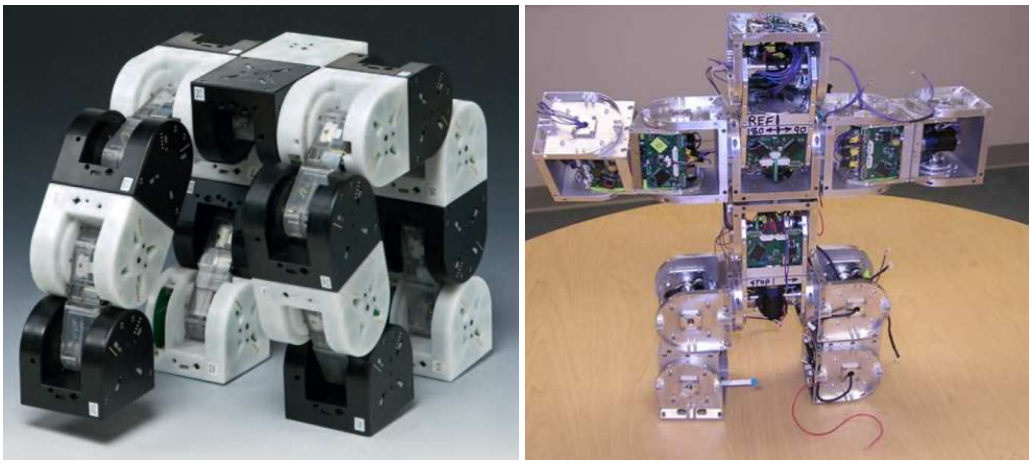

FIGURE 2.5: (a) MTRAN III four legged configuration. Walking occurs with chain-like motions, but reconfiguration occurs with modules at specific lattice positions. (b) a complex assembly of SuperBot modules (Salemi, Moll, and Shen, 2006).

The SUPERBOT system developed by Salemi et al. (Salemi, Moll, and Shen, 2006)

is another example of a hybrid system (Fig 2.5b), which is designed for NASA space exploration programs to remain robust and flexible enough to operate in harsh and uncertain environments. Each module has 3 DoF (two similar to MTRAN with an added twist DoF). The power can be shared within module throught its bonding mechanism. They can communicate via high-speed infra-red light emitting diodes (LED). The modules are controlled using hormone-inspired distributed controllers as developed for the CONRO project.

Another example is Cubimorph (Roudaut et al., 2016), a modular interactive device consists of multiple cubes that accommodates touchscreens on each of the six module faces. The cubes are located in a lattice system, but they use a hing-mounted turntable mechanism in order to change its shape to fit functionalities by following a chain reconfiguration algorithm.

2.2.4 Mathematical tools involved in MSR

Pamecha et al. (Pamecha, Ebert-Uphoff, and Chirikjian, 1997) and Chiang and Chirikjian (Chiang and Chirikjian, 2001) defined a few basic transformations of the modules. Reconfiguration of a MSR system was based on recursively calculating intermediate configurations between a start and a destination configuration, until the transition between consecutive configurations was immediately achievable by the basic moves. The Hungarian method, a combinatorial optimization algorithm that solves the assignment problem in polynomial time (Kuhn, 1955), was applied to obtain optimally matched module pairs between two configurations (For details about Hungarian method, see Appendix B). For each pair, the average coordinates of the modules determined the location of an intermediate module. This approach is applicable to modules of various shapes, such as 2D hexagon (Pamecha, Ebert-Uphoff, and Chirikjian, 1997) and 2D lattice (Chiang and Chirikjian, 2001). The algorithm they introduced inspired another research for PnP sequencing (Pan, Chen, and Dritsas, 2017), which is described in Chapter 3.

2.3 Folding

Gao et al. proposed a new multi-primitive folding framework such as using tetrahedral, cuboidal, prismatic, and pyramidal components, called Kinetogami, to enable design of 3D structures and mechanisms all folded from preplanned printed sheet materials (Gao et al., 2013). It's a combinatorial morphing system by varying the design of individual components in 2D so that one 3D shape can be reconfigured into another by folding.

In the work of Ding (Ding and Lu, 2013), the theory of chain-type modular reconfigurable mechanisms is investigated. The Clifford algebra is employed to represent the chain geometry, and combined with exponential formula for kinematics, discrete motion sequences can be calculated to get the position and orientation of each module. Rubik Snake is used to evaluate the principle of modular reconfiguration mechanisms.

Cheung et al. presents a technique to programmatically general any continuous 2D or 3D shape from 1D strings (Cheung et al., 2011). The technique includes the application of space-filling curve in Euclidean 2D and 3D space, and a subdivision of one cell into 4 and 8 sub-cells. The technique is validated with dynamics simulations as well as experiments with chains of modules that pack on a regular cubic lattice. Using this technique, Tibbits proposed Programmable materials based on Logic gates, called Logic Matter, for large-scale construction by folding (Tibbits, 2012; Tibbits and Cheung, 2012).

Xu et al. proposed a modular chain-like structure of links and connectors nodes which can be folded into various 2D or 3D structures in a lattice system (Xu, Mccann, and Dollar, 2016; Xu, McCann, and Dollar, 2017). The node geometry consists of a diamond-like shape that is one-twelfth of a rhombic dodecahedron, with magnets embedded on the faces to allow a forceful and self-aligning connection with neighboring links. Prototype consisting of 350 links are demonstrated in their work.

2.4 LEGO Construction

A LEGO construction problem was proposed in 1998 (Gower, Heydtmann, and Petersen, 1998): Given any 3D body, how can it be built from LEGO bricks? This problem

is as a prerequisite listed here for the contents of Chapter 4. Nowadays, the problem is largely solved owing to several computational algorithms (Kim, Kang, and Lee, 2014). These algorithms mostly fall into two categories: one focuses on interlayer connectivity and the other on structural stability. For example, the algorithm proposed in (Gower, Heydtmann, and Petersen, 1998) evaluated the connectivity between bricks by heuristics, in which a cost function was defined as the evaluation metric. The algorithm aimed to optimize the cost and automatically generate an assembly. Later, alternative optimization methods, such as an evolutionary algorithm (Petrovic, 2001) and a cellular automata algorithm (Smal, 2008), were introduced to improve to the computational speed and the generated LEGO assembly.

More recent work focused on the balance and stability of LEGO models. In (Ono, Alexis, and Chang, 2013) and (Testuz, Schwartzburg, and Pauly, 2013), an initial LEGO structure was represented by a graph; then a graph algorithm was applied to identify structurally weak points corresponding to weak LEGO bricks; the vicinity of these bricks were replaced by an alternative layout to achieve stronger assembly. The process was iteratively applied till the overall structural strength was up to certain criteria. Luo et al. (Luo et al., 2015) used a force-based metric to evaluate the balance condition of LEGO bricks in terms of the force and torque they were subject to. Weak and unstable bricks were modified to avoid collapsing. Several real-sized objects were constructed to verify the force-based method. Hong et al. (Hong et al., 2016) proposed a centroid adjustment method that can optionally hollow the interior of a LEGO model. Structures produced by the method can stand in a particular pose steadily. Zhang et al. (Zhang et al., 2016) proposed a divide-and-conquer method through a concept called "pseudo floor", which divides a structure into components without floating bricks. The components can be assembled separately and then be connected to each other to produce the final assembly.

All the above methods address the problem of LEGO construction, while Pasek and Yip-Hoi (Pasek and Yip-Hoi, 2005) proposed an interesting computer integrated manufacturing system based on LEGO bricks. They described various components of the system aimed at educational purposes, and ways to assemble and disassemble LEGO models. They even showed conceptual ideas of a gripper designed to pick and place

LEGO bricks automatically. Their work touches upon a research topic has not been explored to the best of our knowledge. That is how to achieve shape transformation from one LEGO model to another. This involves optimization strategies to minimize the steps of transformation and to maximize the reuse rate of LEGO bricks.

Chapter 3

Sequencing for Homogeneous (Voxel) Structures[1]

"Minimalism is not a lack of something. It's simply the perfect amount of something."

— Nicholas,Burroughs

3.1 Overview

Specifically, we study PnP motion planning strategies for a Cartesian robot with at least three axes to transform a set of equally-sized parts from one configuration to another. Our application is relevant to additive manufacturing processes based on stacking but it has broader applications to building construction, rapid prototyping and industrial engineering. There are certain assumptions of the capabilities and limitations of the operating machine such as:
(1) It can reach a 3D work envelope without penetrating a structure so there are no collisions between the parts and the machine;
(2) The parts are held from their top surface using a gripper such as a pneumatic vacuum suction system; and
(3) The parts are spaced apart within tolerance so interface friction characteristics are not regarded during final placement.

[1]The work in this chapter has previously been published in (Pan, Chen, and Dritsas, 2017)

The proposed PnP motion planning has three processing stages, as illustrated in Figure 3.1:

(1) Rasterization,

(2) Model Alignment, and

(3) Motion Sequencing.

The method accepts a closed polygonal mesh as input, which is rasterized into voxels.

3.2 Preprocessing

3.2.1 Rasterization of Pyramidal Structures

Rasterization of a triangle mesh is a well studied topic with many algorithms available in the open literature (Huang et al., 1998; Ix and Kaufman, 2000). We rasterize well-oriented mesh surfaces into voxels followed by a flood-fill operation that assigns interior voxels.

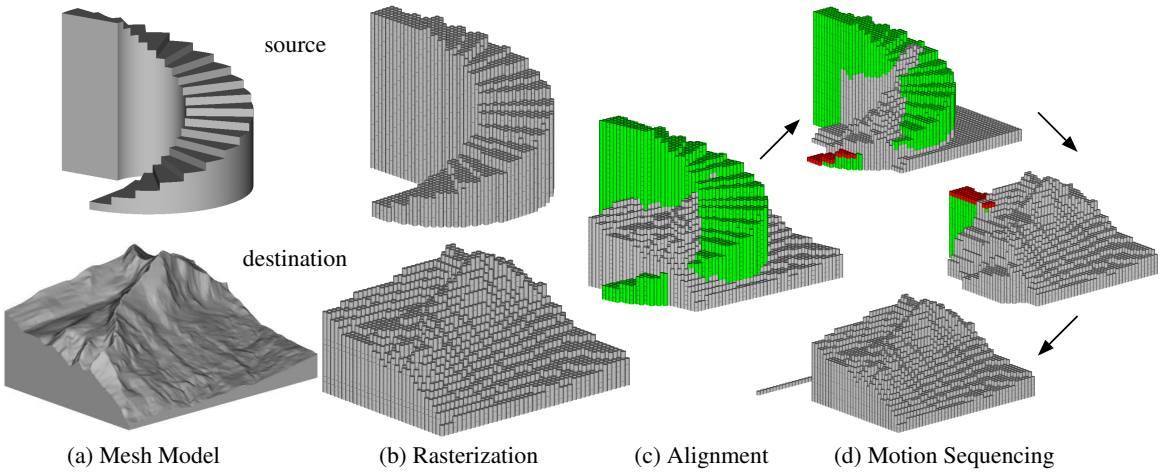

FIGURE 3.1: Processing stages. (a) Input mesh models. (b) Rasterized models. (c) Model alignment. (d) Motion sequence to transform the source to destination.

3.2.2 Model alignment

Model alignment aims to maximize the overlapping volume of two structures and in that respect, minimize required actions for reconfiguration. It is not a trivial problem in computational geometry. Many studies have been conducted in 2D (De Berg et al., 1998; Fukuda and Uno, 2006; Vigneron, 2014). There are also some work in 3D (Fukuda and Uno, 2007; Ahn, Cheng, and Reinbacher, 2013; Ahn et al., 2014) and they deal with shapes in polyhedral representation. Complexity and computational overhead of these methods are considerable. We are dealing with the rasterized representation, and if we assume that the structures are already aligned in the vertical direction on input and that only rigid-body translation without rotation is allowed during alignment, the problem becomes computationally affordable by exhaustive search in the horizontal plane.

After two rasterized structures are aligned, we can classify voxels in three categories (e.g. Figure 3.2).

Mover (M): voxels present only in the source set,

Void (V): voxels present only in the destination set,

Overlapped: voxels present in both sets.

As their names suggest, to transform the source to destination M must be moved to V voxels and the overlapped voxels shall remain in place.

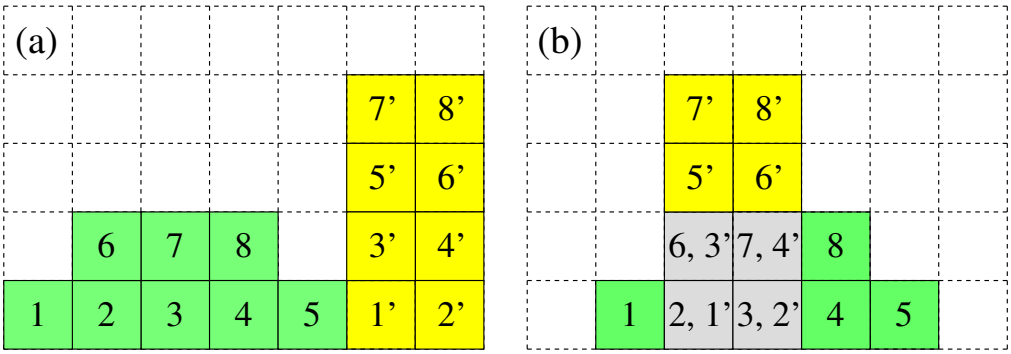

FIGURE 3.2: Model alignment. (a) Unaligned structures. Structure 1 has voxels 1 to 8. Structure 2 has voxels 1′ to 8′ (b) Aligned structures. M voxels in green, V voxels in yellow, and the overlapped voxels in gray.

If a transformation involves an unequal number of M and V, we will put extra M voxels to palette or move additionally required V voxels from palette. The shape of the palette is defined as a linear stack without loss of generality.

3.3 Voxels Sequencing

3.3.1 Cost function

Based on the Cartesian coordinate system, we define the cost of moving an M to a V voxel as

$$c_{ij} = |x_j - x_i| + |y_j - y_i| + 2z_{max} - z_i - z_j \tag{3.1}$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) denote the coordinates of M_i and V_j respectively, and z_{max} is the clearance or rapid motion plane. The cost function models a three-axis Cartesian robotic machine: the movement in the x and y directions is executed sequentially and is only allowed at z_{max} to avoid collision with the structures. As depicted in Figure 3.3, the cost function consists of four distances: d_{zi}, moving M_i to z_{max}; d_x and d_y, moving M_i in the z_{max} plane so that its x and y coordinates coincide with those of V_j; d_{zj}, moving M_i from z_{max} to V_j.

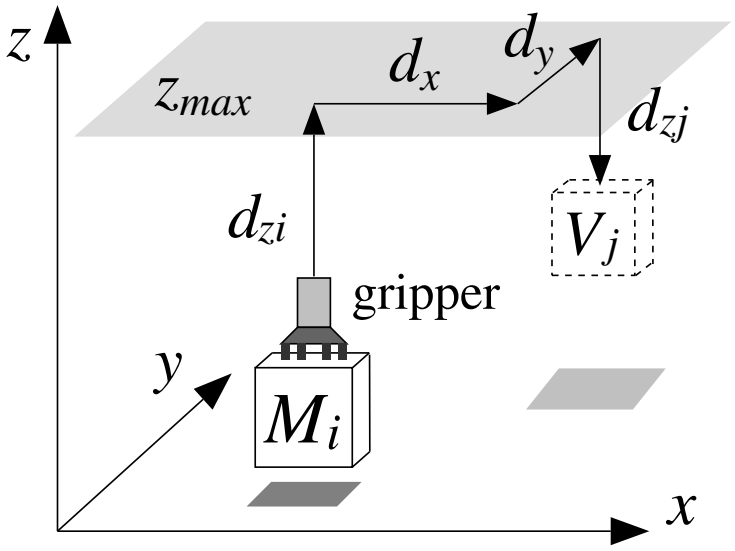

FIGURE 3.3: Cost of PnP. A typical move of M_i to V_j consists of four distances: d_x, d_y, d_{zi} and d_{zj}.

3.3.2 Mapping M to V

To generate a motion sequence, we are concerned with which M voxels are moved to which V voxels. There exist many ways of mapping M to V while we are interested in

the one that produces the lowest total cost of movement. Assuming that there are n M to be moved to the same number of V, we can calculate the costs of moving each M to all V by Equation 3.1. The costs can be expressed in a cost matrix

$$C_{n\times n} = \begin{array}{c} \\ M_1 \\ M_2 \\ \vdots \\ M_n \end{array} \begin{array}{c} \begin{array}{cccc} V_1 & V_2 & \cdots & V_n \end{array} \\ \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix} \end{array} \tag{3.2}$$

where the i-th row contains the costs of moving M_i to all V, and the j-th column contains the costs of moving all M to V_j. The cost matrix completely captures every possible mapping from an arbitrary M to an arbitrary V. Next we find the minimal total cost such that each M is mapped onto a distinctive V.

In principle, finding the best mapping between M and V is an assignment problem, which is a combinatorial optimization problem having factorial complexity to the number of elements (Burkard, Dell'Amico, and Martello, 2009), stated in the next section (Burkard, Dell'Amico, and Martello, 2009). A brute-force search for the minimal cost requires comparison of $n!$ sequences, while the Hungarian method is a much more efficient search algorithm with time complexity $O(n^3)$ (Kuhn, 1955; Frank, 2005). A standard numerical procedure of the Hungarian method feeding the cost matrix as the input is applied. For further information about the implementation Hungarian method, please refer to Appendix B. The output is n pairs of M and V with the total cost of moving each M to the corresponding V guaranteed to be minimum. In addition, computing a solution requires consideration of constraints that originate in physical motion limitations such as the machine's reach, the geometric shape of the elements, and potential collisions among the machine and elements.

3.3.3 Assignment Problem

The assignment problem is a special type of transportation problem where the objective is to minimize the cost of allocating a number of jobs to a number of persons so

that one person is assigned to only one job. The assignment model is useful in solving problems such as assignment of machines to jobs, assignment of salesmen to sales territories, etc. It may be noted that with n persons and n jobs, there are $n!$ possible assignments. The direct way to find the optimal assignment is to consider all the $n!$ possible arrangements, compute their total costs, and select the assignment with minimum cost. However, due to exponential computational cost, this method is not suitable. For solving assignment problem, there is an efficient method which was developed by a Hungarian Mathematician D.König.

3.3.4 Physical constraint

Despite that the lowest-cost M-V pairs can be generated by the Hungarian method, they do not necessarily lead to feasible sequences, where feasibility is determined by physical constraints of a PnP process. In this study, our PnP machine uses a pneumatic vacuum suction gripper; so it is subject to a top-access constraint, meaning PnP is only applicable to voxels at the top surface of a structure.

We use shape transformation in Figure 3.2(b) to show examples of a feasible and an invalid sequence. The M: $\{1, 4, 5, 8\}$ are to be moved to V: $\{5', 6', 7', 8'\}$; hence, the cost matrix is

$$C_{4\times4} = \begin{array}{c} \\ 1 \\ 4 \\ 5 \\ 8 \end{array} \begin{array}{c} \begin{array}{cccc} 5' & 6' & 7' & 8' \end{array} \\ \begin{pmatrix} 7 & 8 & 6 & 7 \\ 8 & 7 & 7 & 6 \\ 9 & 8 & 8 & 7 \\ 7 & 6 & 6 & 5 \end{pmatrix} \end{array}. \tag{3.3}$$

The Hungarian method produces the following lowest-cost pairs: $1 \to 7'$, $4 \to 8'$, $5 \to 5'$, $8 \to 6'$, with a total cost $6+6+9+6 = 27$. A feasible sequence is $5 \to 5'$, $1 \to 7'$, $8 \to 6'$, $4 \to 8'$, as illustrated in Figure 3.4. A voxel at the top surface of the structures is picked and placed at each step.

An alternative lowest-cost solution of the Hungarian method is $5 \to 5'$, $1 \to 7'$, $8 \to 8'$, $4 \to 6'$. (The total cost is also 27.) After two steps, it is impossible to move 8 to 8′ and 4 to 6′ because of a deadlock under the top-access constraint shown in Figure 3.5.

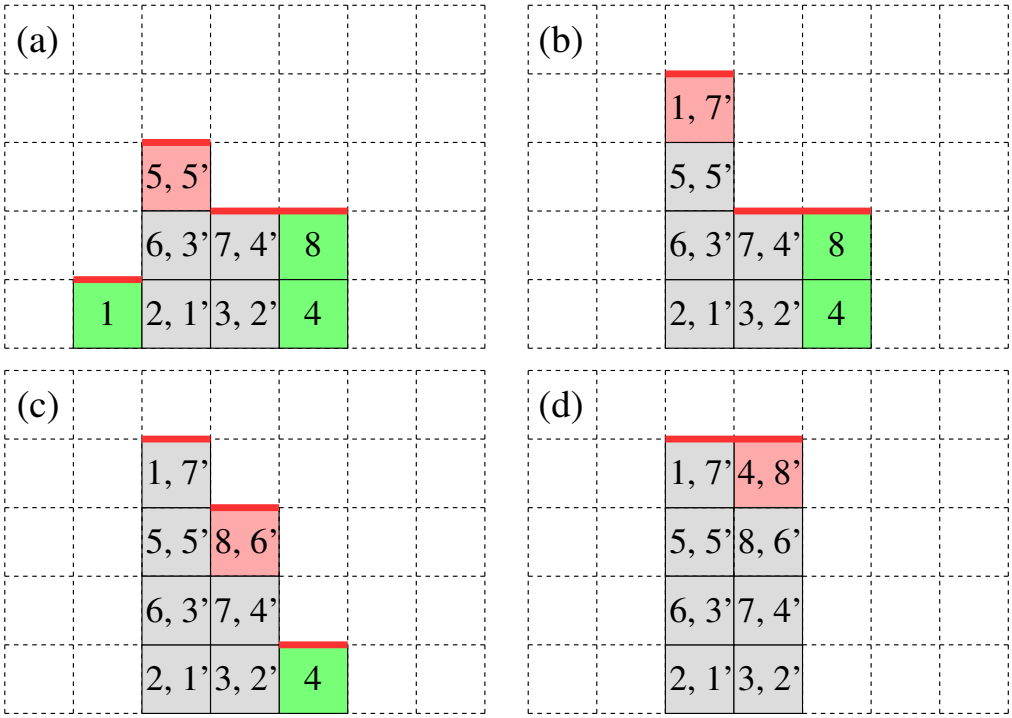

FIGURE 3.4: A feasible sequence of shape transformation in Figure 3.2(b). (a) $5 \rightarrow 5'$, (b) $1 \rightarrow 7'$, (c) $8 \rightarrow 6'$, (d) $4 \rightarrow 8'$. The top surface is indicated in red.

3.4 Strategies

Four strategies: Global Optimization, Local Optimization, Greedy Selection, and Random Selection strategies, are proposed in generating motion sequences.

3.4.1 Global Optimization Strategy (GOS)

The cost matrix contains all M and V, and the resultant M-V pairs from the Hungarian method are guaranteed to produce the minimum total cost; however, the M-V pairs may not be feasible for PnP under the top-access constraint. Hence, GOS is used as a lower bound of the total cost, not as a method for generating valid motion sequences.

3.4.2 Local Optimization Strategy (LOS)

LOS aims at producing motion sequences that are valid against the top-access constraint apriori without computation of collisions by simulation. It thus constructs a cost matrix of M and V directly accessible on the respective structures and applies the

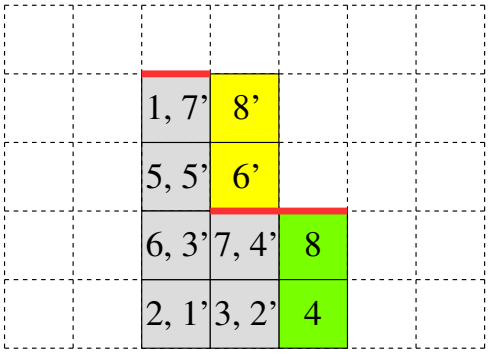

FIGURE 3.5: Invalid sequence: $8 \rightarrow 8'$ and $4 \rightarrow 6'$, due to deadlock.

same Hungarian method to find the lowest-cost M-V pairs, which is a subset of all M and V; therefore, it is not global but local optimization. After these M are moved to the corresponding V, the top voxel surface M and V of each structure are updated. Then, a new cost matrix is calculated based on the new set of M and V. The process is repeated until all M are moved to V.

For example, in Figure 3.2(b) at the start of transformation, the top-surface M and V are $\{1, 5, 8\}$ and $\{5', 6'\}$ respectively. (V $\{7', 8'\}$ are on the top surface at the final stage, not the initial stage.) The cost matrix is

$$C_{3\times2} = \begin{array}{c} \\ 1 \\ 5 \\ 8 \end{array} \begin{array}{c} \begin{array}{cc} 5' & 6' \end{array} \\ \begin{pmatrix} 7 & 8 \\ 9 & 8 \\ 7 & 6 \end{pmatrix} \end{array}. \tag{3.4}$$

The number of M is larger than that of V; the Hungarian method is able to discard an M associated with high costs such that the total cost of paired M and V is minimum. The M-V pairs found are $1 \rightarrow 5'$ and $8 \rightarrow 6'$ while $M\{5\}$ dose not participate in transformation at this stage. PnP may be applied to these pairs in any order [Figure 3.6(a)]. Next, the top-surface M and V become $\{4, 5\}$ and $\{7', 8'\}$ respectively and the new cost matrix is

$$C_{2\times2} = \begin{array}{c} \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cc} 7' & 8' \end{array} \\ \begin{pmatrix} 7 & 6 \\ 8 & 7 \end{pmatrix} \end{array}. \tag{3.5}$$

The M-V pairs are $4 \rightarrow 7'$ and $5 \rightarrow 8'$ [Figure 3.6(b)], or $4 \rightarrow 8'$ and $5 \rightarrow 7'$. The total cost of LOS is $7 + 6 + 7 + 7 = 27$, which happens to be the lower bound produced by GOS (Section 3.3.4).

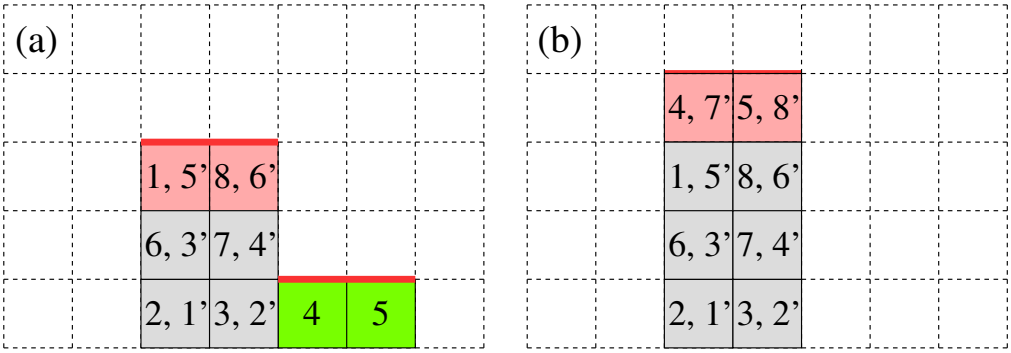

FIGURE 3.6: Motion sequence generated by LOS. (a) First stage. (b) Second stage.

3.4.3 Greedy Selection Strategy (GSS)

GSS aims at producing valid motion sequences without using the Hungarian method. The cost matrix is constructed in exactly the same way as LOS. The lowest-cost M-V pair in the matrix is selected by simply searching for the first instance of the lowest cost value element. The M is moved to the paired V, and their corresponding row and column are removed from the matrix. This completes one step of transformation.

If new M and V appear on the top surface as a consequence the above transformation, the cost matrix is updated by adding a row for an M and a column for a V. Then, the same process is applied: searching for the lowest-cost M-V pair in the matrix, moving the M to V, and removing their corresponding row and column, etc. The process is repeated until all M are moved to V. The strategy is named greedy move because at each step only the lowest-cost M-V pair is moved, not all those on the top surface.

Based on the example of Figure 3.2(b), GSS produces motion sequence: $8 \rightarrow 6'$, $4 \rightarrow 8'$, $1 \rightarrow 5'$, $5 \rightarrow 7'$, shown in Figure 3.7. The total cost is $6 + 6 + 7 + 8 = 27$.

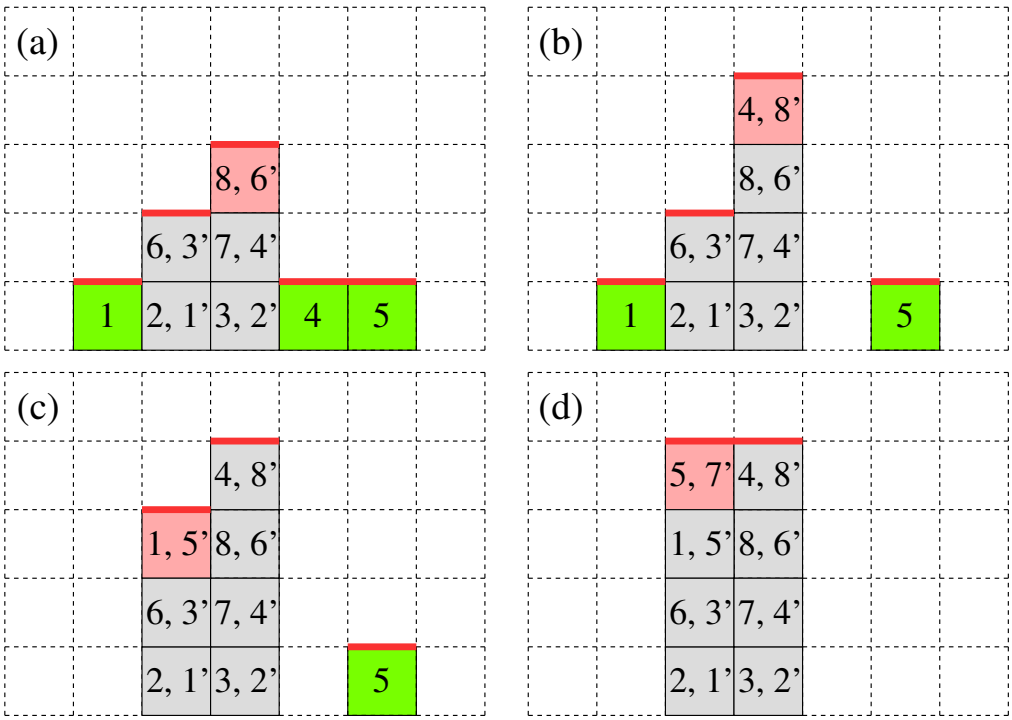

FIGURE 3.7: Motion sequence generated by GSS. (a) $8 \rightarrow 6'$. (b) $4 \rightarrow 8'$. (c) $1 \rightarrow 5'$. (d) $5 \rightarrow 7'$.

3.4.4 Random Selection Strategy (RSS)

RSS is similar to GSS except that at each step a random, as opposed to a lowest-cost, M-V pair is selected for PnP. Thus processing time for searching the top-voxel surface for the best element is eliminated. RSS simulates a relatively random PnP process and its total cost may be considered as an average of all different feasible motion sequences.

3.5 Results and Discussions

The motion planning strategies were tested on several computer models shown in Figure 3.8. The models were first rasterized into structures of similar sizes, regardless of their original dimensions. Due to the rasterization method implemented, it's hard to control accurately the number of voxels in a rasterized model; hence, shape transformation required moving material from and to a linear stack. Table 3.1 shows the total cost and computational time of the strategies in application to transforming the models to each other. Models on the left column are the source and those on the top row are

the destination of transformation. The number below a model indicates the number of voxels of the rasterized structure. The total cost is a representation of the workload when shape transformation is realized in a physical PnP system. The computational time is the CPU processing time for generating a motion sequence from a C++ program running on a Windows 64-bit PC, Intel(R) Core(TM) i3-2310M CPU 2.10 GHz, RAM 8 GB.

For each transformation shown in Table 3.1, it is the number of M-V pairs, not the number of voxels in the structures, that suggests the amount of work needed because prior to PnP, the structures have been aligned and overlapped voxels do not require movement. This can be seen in the transformation from the pyramid (left column) to wall (top row), and from the pyramid to igloo (top row). The first transformation involved 1643 M-V pairs while the second one only involved 325 M-V pairs. The pyramid and igloo are more similar in shape, thereby better aligned, which is the reason for the reduction in the cost and time.

Overall, in terms of the cost metric, GOS is the best strategy of the four as it guarantees to produce the lower bound of the total cost; however, the M-V pairs generated by GOS do not suggest a motion sequence by default. Sometimes, they do not lead to a feasible motion sequence at all (Section 3.3.4). The second-best strategy is LOS, which generates groups of M-V pairs in several steps of processing. The groups naturally determine a sequence. For M-V pairs within each group, they can be moved in any sequence as they are all on the top surface of the respective structures. GSS is the third best strategy, slightly worse than LOS but clearly better than RSS.

In terms of the time metric, RSS is the fastest strategy since little calculation is needed. All it does is to update the top-surface M-V pairs of two structures, and pick a random pair. GSS and LOS are similar in the speed of processing. GSS has to update the top-surface M-V pairs after every PnP action; therefore, albeit light-weight processing at each step, the accumulated processing time may not be shorter than that of LOS. LOS applies the Hungarian method multiple times; at each time a group of M-V pairs, not one pair, are completed with PnP; hence, the number of iterations of calculation is much fewer than that of GSS, and the cost matrix is not huge. In contrast, GOS only applies the Hungarian method once to all M and V with a huge cost matrix, which

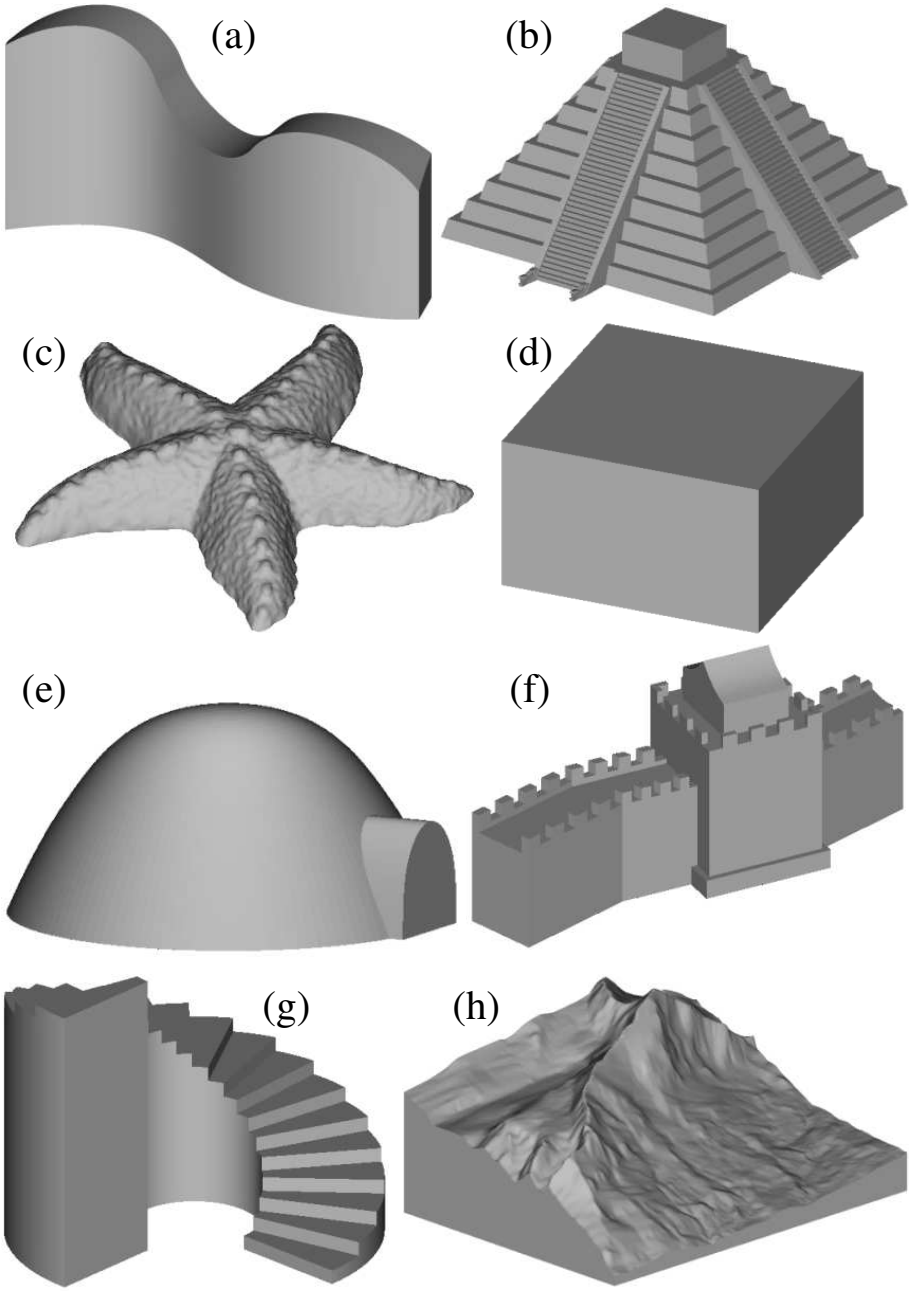

FIGURE 3.8: Test models: (a) wall, (b) pyramid, (c) starfish, (d) cube, (e) igloo, (f) castle, (g) stairs, and (h) Eiger mountain.

makes it the most time-consuming strategy.

The performance of the strategies with respect to different resolutions of rasterization is also studied in this work. Figure 3.9 shows the results based on a particular transformation: from the starfish to the castle model, while other transformations exhibit similar patterns. As can be seen, the total cost increases about linearly with the

increment in the number of M-V pairs. Same applies to the time with exception of GOS, whose time complexity is $O(n^3)$. The results of GOS at high resolutions are not included in Figure 3.9(b) to avoid drastic downscaling of the time axis. Most importantly, the overall trend shows that increasing resolution will not induce cubic increase in the time of LOS, whereas one might thought so intuitively for it is based on the Hungarian method with time complexity $O(n^3)$.

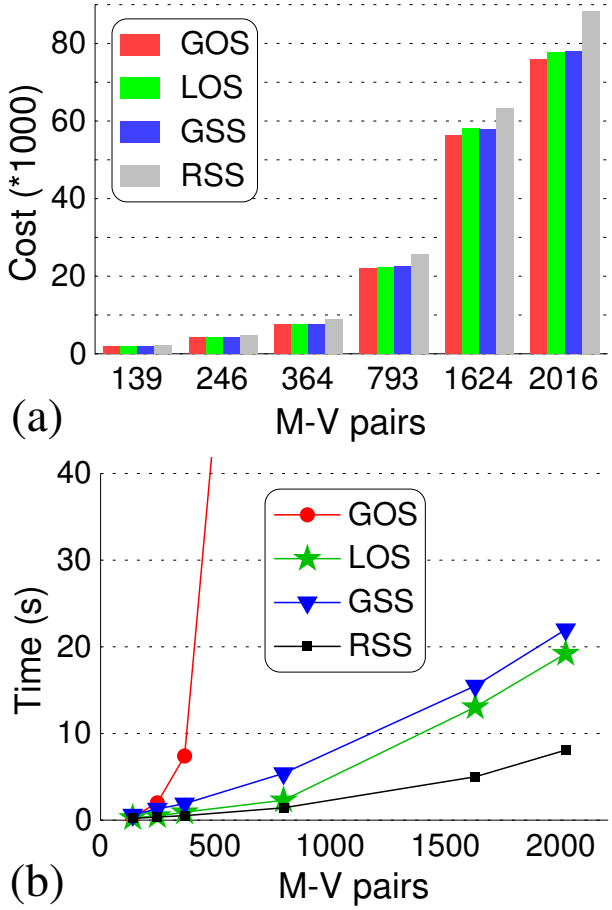

FIGURE 3.9: Total cost (a) and computational time (b) of the strategies based on transformation from the starfish to the castle model at different resolutions of rasterization. The number of M-V pairs indicates the resolution: more M-V pairs, higher resolution.

Based on the comparisons, LOS stands out as it can achieve an appealing balance between the cost and processing time. Figure 3.10 shows a sequence of transformation obtained by LOS in the order of wall, pyramid, starfish, cube, igloo, castle, stairs, and Eiger. The resolution of rasterization is set to show clearly each model and is much

higher than that used in the previous experiments. Note that in several transformations material were moved from and to a linear stack to compensate for unequal number of M and V.

In our study, the principle and experiments suggest that an optimal motion sequence can only be generated in consideration of physical constraints. Although the top-access constraint is a specific assumption and it prevents the construction of overhanging structures, it's anticipated that the proposed approach in LOS has more general applications. Under different situations, alternative cost functions and physical constraints may be framed; the approach of first selecting voxels satisfying the constraints, then optimizing M-V pairs would lead to methods that are computationally light weight and produce low-cost solutions.

3.6 Conclusion and Future Work

Based on a particular PnP model, several strategies have been proposed to achieve shape transformation of rasterized 3D structures. The global strategy GOS produces the lower bound of the total cost but does not guarantee a feasible motion sequence. Comparison of different strategies shows that the local strategy LOS outperforms the others that are able to generate feasible PnP sequences.

Application of the proposed methods can be found in areas such as collective construction, automatic assembly, reconfigurable robots, and rapid prototyping. In a large-scale problem involving the movement of thousands or more parts, minimizing the workload and obtaining a motion sequence in a short time are both important. The strategy proposed in this paper is a reference for alternative methods under other application-dependent physical constraints.

The top-access constraints restrict LOS to only non-overhang structures, as shown in Figure 3.10. However, LOS is still extensible with given supportive materials, for instance, sticky structures or bricks which can buckle each other like Lego bricks enable the overhanging shape. In these cases, the conditions will be more complicated considering the order of motion planning. Our future work will focus on algorithms to transform structures made of several types of parts, e.g. Lego. Algorithms based on parts

with different properties, such as color and texture can be another interesting topic. This would enable transformation of structures with varying surface features. Modeling a realistic robotic machine is another topic to investigate. A more flexible robotic machine may save the trouble to move all the way to the clearance plane. Perhaps, the topology of a structure can indicate which part should be moved first. Algorithms that combine topological information with machine capabilities may lead to optimization strategies that could solve the problem of accessibility and motion sequence in one framework.

TABLE 3.1: Comparison of the total cost and computational time.

From	To		wall 2435	pyramid 2455	starfish 2550	cube 2250	igloo 2346	castle 2441	stairs 2437	Eiger 2466
wall 2435	M-V pairs			1643	2248	1625	1677	2153	1876	1581
	Cost	GOS		60021	89191	54855	58875	91192	71779	46845
		LOS		60199	89263	55789	59479	91192	71917	48038
		GSS		61201	89377	56585	60509	92536	74059	47883
		RSS		62413	98707	58653	62419	93630	74931	51523
	Time	GOS		4440	9431	3744	4058	10718	4621	1684
		LOS		9.5	21.1	9.9	9.5	14.9	11.2	3.6
		GSS		15.1	23.4	15.1	15.4	19.9	18.9	14.2
		RSS		5.8	8.9	6.0	6.0	9.2	7.0	2.2
pyramid 2455	M-V pairs		1643		1492	1191	325	1748	1272	1072
	Cost	GOS	60021		55470	39384	8911	62229	39336	30259
		LOS	60761		55514	39384	8977	62241	39472	30679
		GSS	62009		55840	39470	9433	62791	40016	30323
		RSS	62413		56664	39756	9809	64971	43168	31079
	Time	GOS	1093		1829	1147	6.1	4097	1333	164
		LOS	9.1		22.3	8.9	2.6	14.7	7.3	2.3
		GSS	15		14.6	10.5	2.5	16.4	11.4	9.4
		RSS	6.4		6.0	4.8	1.7	7.0	5.2	1.5
starfish 2550	M-V pairs		2248	1492		2169	1552	2016	1895	1846
	Cost	GOS	89191	55470		73674	51677	75891	88120	71114
		LOS	89351	55510		73682	51677	76065	88134	71628
		GSS	89649	55888		74074	51939	77929	88502	71376
		RSS	98691	56596		75826	52735	88209	90546	72074
	Time	GOS	2677	2757		10690	3108	5137	5436	3349
		LOS	15.8	22.5		30.2	30.3	19.2	13.9	8.1
		GSS	23.7	14.7		23.7	15.4	21.8	19.6	20.1
		RSS	9.0	5.7		8.9	6.0	8.1	7.7	2.7
cube 2250	M-V pairs		1625	1191	2169		973	1590	1141	902
	Cost	GOS	54855	39384	73674		26625	59400	34446	22184
		LOS	56011	39384	73728		26625	59724	34678	22794
		GSS	56627	39460	73862		26667	59594	35508	22414
		RSS	58607	39792	75942		27017	61196	39372	26158
	Time	GOS	1519	758	6248		551	3304	596	92.3
		LOS	9.6	9.0	26.0		6.0	9.0	6.5	1.8
		GSS	14.7	10.4	23.3		8.1	14.4	9.7	7.6
		RSS	6.5	4.9	9.4		4.1	6.6	4.8	1.3
igloo 2346	M-V pairs		1677	325	1552	973		1703	1141	964
	Cost	GOS	58875	8979	51677	26625		62783	34432	24789
		LOS	60211	9047	51683	26625		62861	34754	24967
		GSS	60949	9411	51887	26649		63215	34848	25093
		RSS	62367	9809	52867	27025		66747	39526	26457
	Time	GOS	1319	7.2	2341	382		3714	732	81.6
		LOS	9.9	2.6	30.4	6.2		13.9	6.8	1.9
		GSS	15.1	2.4	15.3	8.0		15.8	10.0	8.1
		RSS	7.0	1.8	6.7	4.0		7.7	5.1	1.3
castle 2441	M-V pairs		2153	1748	2016	1590	1703		1358	1839
	Cost	GOS	91192	62229	75891	59400	62783		45249	74994
		LOS	91192	62267	76255	59762	62855		45679	74780
		GSS	92546	62585	77831	59592	63007		46229	75050
		RSS	93630	64971	88217	61232	66711		47431	75638
	Time	GOS	3502	4832	4376	1478	3737		808	1004
		LOS	13.6	17.2	20.8	9.4	15.6		8.3	4.6
		GSS	19.6	16.6	21.2	14.4	16.3		11.9	18.9
		RSS	9.6	8.1	9.6	7.5	8.1		6.6	2.5
stairs 2437	M-V pairs		1876	1272	1895	1141	1141	1358		1384
	Cost	GOS	71779	39336	88120	34446	34432	45249		47449
		LOS	71823	39578	88328	34664	34722	45527		47511
		GSS	75887	39698	88198	35640	34624	45897		48289
		RSS	74931	43168	90554	39320	39426	47431		50487
	Time	GOS	1513	1209	3699	603	591	1121		377
		LOS	12.2	8.5	14.9	7.5	7.1	8.9		3.5
		GSS	17.6	11.1	19.4	9.6	9.9	13.5		11.8
		RSS	8.8	6.3	9.5	5.6	5.7	7.0		1.8
Eiger 2466	M-V pairs		1443	1072	1846	902	964	1839	1384	
	Cost	GOS	50734	30259	71114	22184	24789	74994	47449	
		LOS	51584	30291	71126	22652	24987	74780	47497	
		GSS	51996	30355	71180	22326	24983	75020	48213	
		RSS	52606	31079	72114	26268	26649	75638	50487	
	Time	GOS	2795	215	1145	115	144	3605	716	
		LOS	9.2	2.8	8.6	2.1	2.3	5.6	4.0	
		GSS	14.5	9.3	20.3	7.6	8.6	17.8	12.0	
		RSS	6.0	1.5	2.8	1.4	1.4	2.6	1.8	

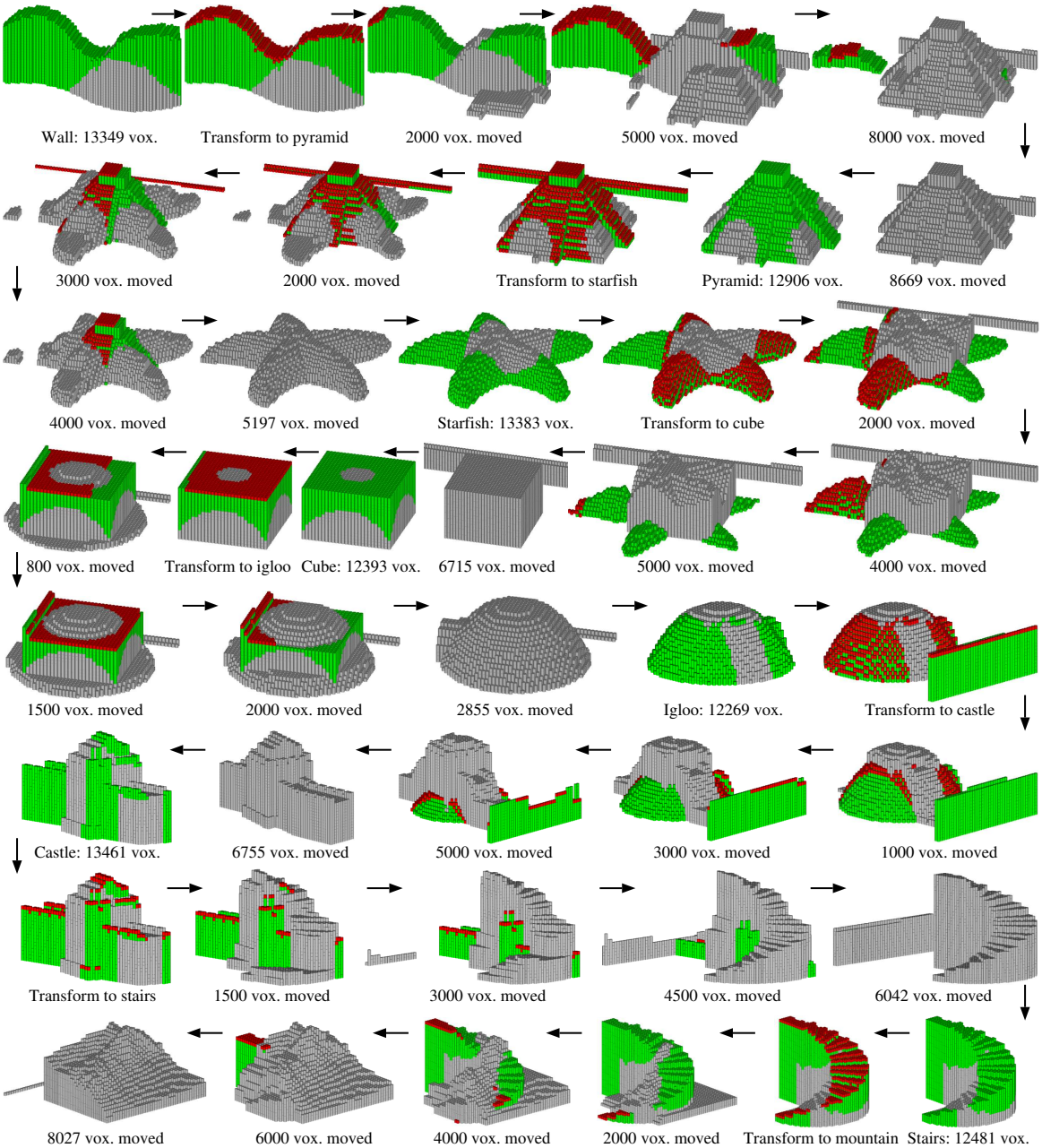

FIGURE 3.10: A sequence of transformation obtained by LOS. M voxels subject to PnP at the current stage are indicated in red. Other M voxels are indicated in green. Voxels of the destination are indicated in gray.

Chapter 4

Sequencing Optimization for Heterogeneous (Brick) Structures

"The fewer moving parts, the better." "Exactly. No truer words were ever spoken in the context of engineering."

— Christian Cantrell, Containment

4.1 Overview

Efficiency is crucial to all kinds of physical construction. In different scenarios, the meaning of efficiency varies. This paper presents a study of efficiency, in terms of cost of motion and material reuse, regarding shape transformation of LEGO models.

The general pipeline is shown in Figure 4.1. The algorithm takes two polygonal mesh models as input. The first model is source, converted (Legolized) into a pre-built LEGO assembly; the second model is the target model. Finally, users could build the second model layer by layer by following the bricks transporting sequences.

4.2 Preprocessing

4.2.1 Rasterization of 3D Model with Color Information

In digital 3D models, color information usually stored in mesh as vertex color, face color and texture color. In the first and second case, color information can be directly retrieved from the file. However, in the last case (more often than not), color is integrated as a bitmap image mapping to the mesh, which is represented as texture color.

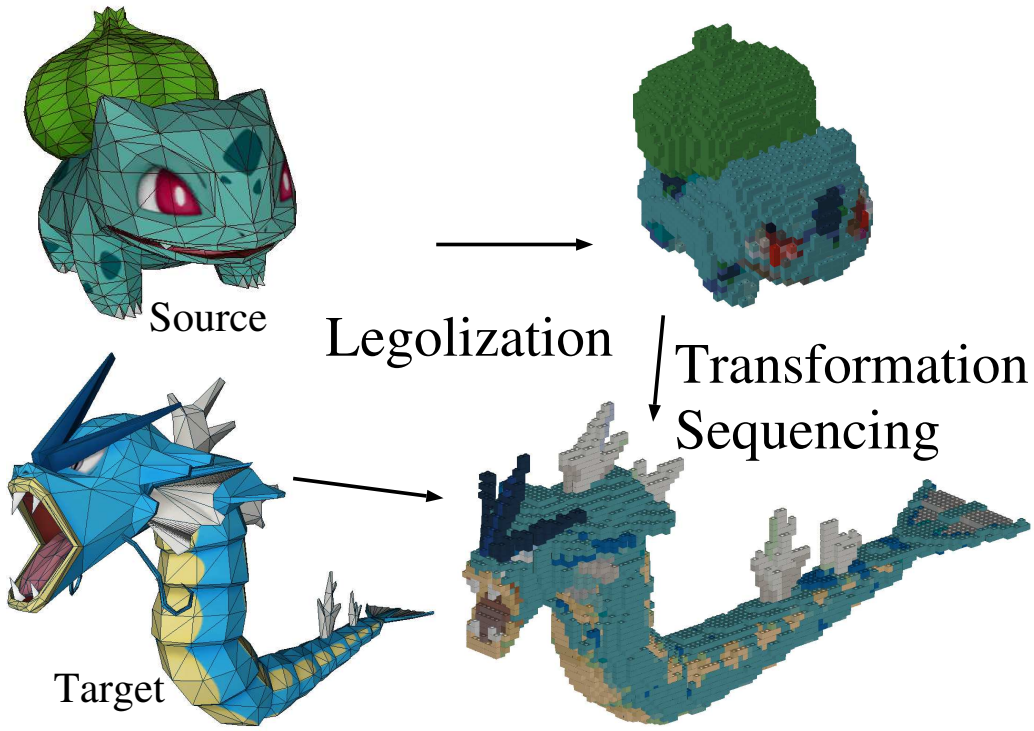

FIGURE 4.1: A general pipeline for transformation of two 3D model (LEGO).

Tools exist which provide functions to transfer texture color into face color or vertex color (Cignoni et al., 2008). To maintain the resolution of color information, an operation of regional subdivision of polygons (eg. triangles) is usually necessary, as shown in Figure 4.4.

A LEGO package has bricks in various shapes and colors. To facilitate the presentation of the principle, seven different shapes and one hundred and fourteen colors are used as shown in Figure 4.2. Then the face color is rounded to the closest standard LEGO brick color in the rasterization process.

FIGURE 4.2: Seven different-shaped LEGO bricks used in this study.

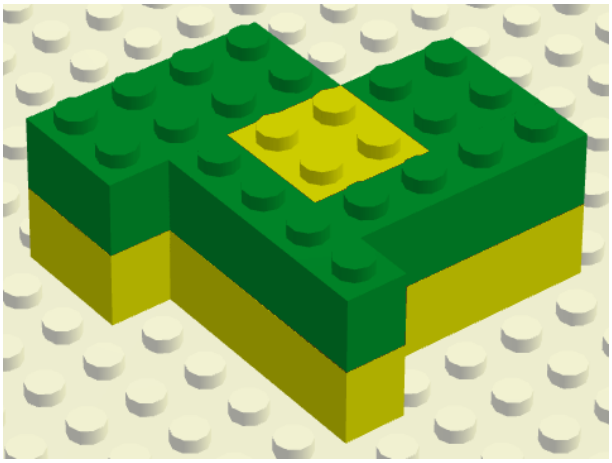

FIGURE 4.3: A fully surrounded brick, the 2-by-2 yellow brick, is not movable.

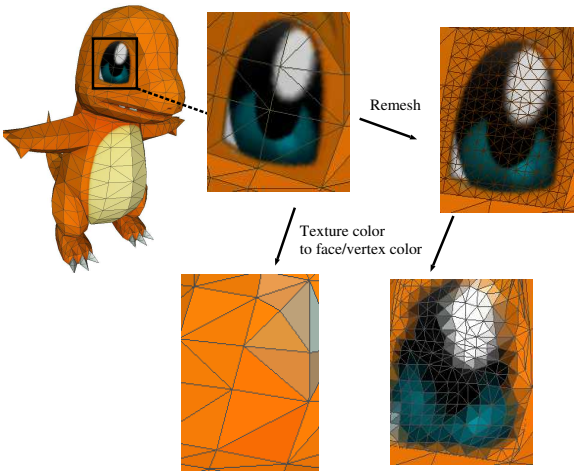

FIGURE 4.4: An operation of regional subdivision of polygons is usually necessary for the translation from texture color to face color.

4.2.2 Legolization: Brick Layout Generation

To generate a LEGO representation of a 3D model, the work of Hong et al. is referred in this section (Hong et al., 2016). The brick layout assembly is generated layer by layer. In each layer, the first step is to select the voxel that has fewer neighbors to merge into a brick, as a voxel with fewer neighbors is more likely to form weak brick connections. The possibility whether a voxel is to be selected is determined by Equation 4.1.

$$P = \begin{cases} 1, & \text{if } nNeighbor <= 3 \\ 1/\exp(nNeighbor), & \text{otherwise} \end{cases} \tag{4.1}$$

The second step is to merge the voxel we selected above into a brick with its mergeable neighbors. According to the heuristic rules proposed by Gower (Gower, Heydtmann, and Petersen, 1998), bricks with more perpendicular connections result in better stability of the sculpture. A simple conclusion is larger bricks commonly own more connections. Therefore, the brick layout of the bottom layer will be merged based simply on the bricks size. For the other layers, the brick merging will be determined by both connections with bricks in the previous layer and its brick size. A brick value integrated the two factors is computed by Equation 4.2, the brick with the best brick value will be selected thus merged.

$$V_{\text{Brick}} = \alpha C_{\text{Connection}} + (1 - \alpha) C_{\text{BrickSize}} \tag{4.2}$$

The value of α ranging from 0 to 1 determines the weight of $C_{\text{Connection}}$ and $C_{\text{BrickSize}}$. Higher α strengthens the connections while increases the demand for big bricks. α = 1 is chosen in this work by assuming an unlimited bricks stock. It should be noted that some structures exist parts that are inevitably pendant when it's assembled bottom-up and layer-by-layer until a type of larger brick, e.g. 2×10 brick is introduced, another alternative solution is to apply the method proposed by Zhang (Zhang et al., 2016). However, bricks like this are still able to connect with bricks in an upper layer. In this case, the $C_{\text{Connection}}$ is 0, we continue to merge it based on brick size.

It should be noted that PnP of a LEGO brick is subject to physical constraints. A brick fully or partly under other bricks cannot be picked; a brick disconnected from a partially built model cannot be placed as it would be hanging in the air. Figure 4.3 shows an additional constraint. Although the 2-by-2 yellow brick is on the top surface, it cannot be picked without moving the surrounding green bricks; hence, such fully surrounded bricks cannot be picked either. Hereafter, bricks that satisfy the physical constraints are regarded as movable bricks.

4.3 Bricks Sequencing

Two basic approaches are proposed. The first approach assumes that a source and a target LEGO models are available. A PnP motion sequence to transform the source to target is generated by an algorithm. The second approach assumes that the source LEGO model is available while the LEGO representation of the target is not fixed but is created from an interim voxelized model during transformation. Figure 4.5 depicts the difference between the two approaches: rigid transformation (RT) and flexible transformation (FT). In both approaches, the source model is disassembled from top to bottom and the target model is built from bottom to top, layer by layer. In RT, without loss of generality a variant of the algorithm described in (Hong et al., 2016) is applied to generate both LEGO models; at each layer, large LEGO bricks are used with priority. In FT, movable LEGO bricks on the source model are used with priority. When required bricks are unavailable, they are obtained from a stock that is assumed to have unlimited supply of all types of bricks; then large bricks are use with priority just as in RT. The motivation behind FT is to increase the reuse rate; however, the side effect is that the target model generated by FT has larger number of LEGO bricks and is less stable than that by RT. While the stability issue is not concerned in this study, the metrics of performance such as the cost of shape transformation associated with the number of LEGO bricks are evaluated.

In the previous section, the polygonal mesh is converted to a rasterized model made of identical voxels, which is described in Chapter 3. Then, the voxels are grouped into LEGO bricks, which is the key step in LEGO generation. The algorithm described in (Hong et al., 2016) generates various possible voxel groups (i.e. LEGO bricks) at each layer by heuristics, and applies a scoring system to determine the final bricks. The scoring system show as follows is used to generate LEGO models, which is not fixed and can be changed according to different situations.

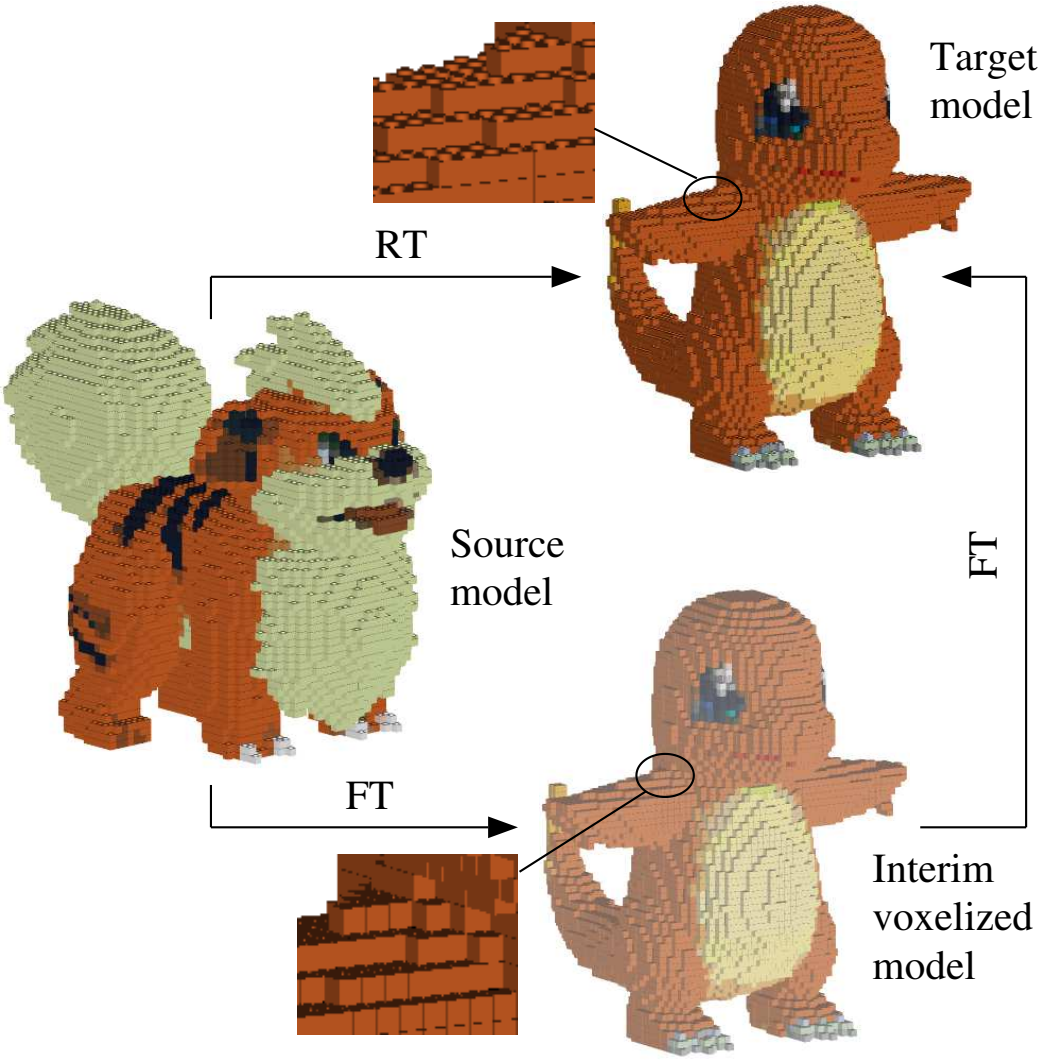

FIGURE 4.5: Two basic approaches: rigid transformation (RT) and flexible transformation (FT).

Rigid Transform Approach (RT) In the RT approach, the score of a brick b is calculated by

$$S_1 = \begin{cases} A_b, & \text{Bottom layers} \\ N_c, & \text{Upper layers} \end{cases} \tag{4.3}$$

where A_b is the area of the brick (e.g. $A_b = 4$ for a 2-by-2 brick), and N_c is the number of bricks at a lower layer that b is connected to. LEGO generation goes from bottom to top. A bottom layer does not have a layer beneath it; this includes a base layer and the lowest layer of an overhung section. On a bottom layer, a large brick gets a high score;

on upper layers, a brick with many connections gets a high score.

Flexible Transform Approach (FT) In the FT approach, the source model is generated in exactly the same way as in RT. For the target model, the score of a brick b is calculated by

$$S_2 = \begin{cases} N_b S_1, & N_b > 0 \\ S_1/V & N_b = 0 \end{cases} \tag{4.4}$$

where N_b is the number of movable b type bricks currently available from the source, and V is the number of voxels in the target model. N_b works as a weighting factor that biases towards a type of brick that has abundant supply from the source. If the b type is unavailable from movable bricks ($N_b = 0$), its score is S_1/V, effectively the same as switching back to Equation 4.3. Dividing V ensures that movable bricks always get a higher score than unavailable bricks.

Based on the calculated scores, voxel groups are converted to LEGO bricks in a score-descending order. In general, LEGO models generated by Equation 4.3 is more robust, whereas the target models generated by Equation 4.4 would make use of more bricks from the source model.

4.3.1 Transform Strategies

Under each approach, three strategies were investigated, as illustrated in Figure 4.6. In Strategy 1, the source model is disassembled completely and the LEGO bricks are put to a buffer; then the target model is built using bricks from the buffer and a stock with the buffer used by default. If certain brick cannot be found in the buffer, it is obtained from the stock, which can supply of all types and colors of bricks.

In Strategy 2, movable bricks from the source model are directly picked and placed on the target model. If some required bricks on a target layer do not have matching movable bricks on the source, an initially empty buffer is searched for such bricks. If they exist, PnP continues from the buffer to target; otherwise, one layer of bricks are disassembled from the source and placed to the buffer so that a new layer can be uncovered on the source model. If movable bricks on the new layer satisfy the requirement of the target layer, PnP can resume from the source to target; otherwise,

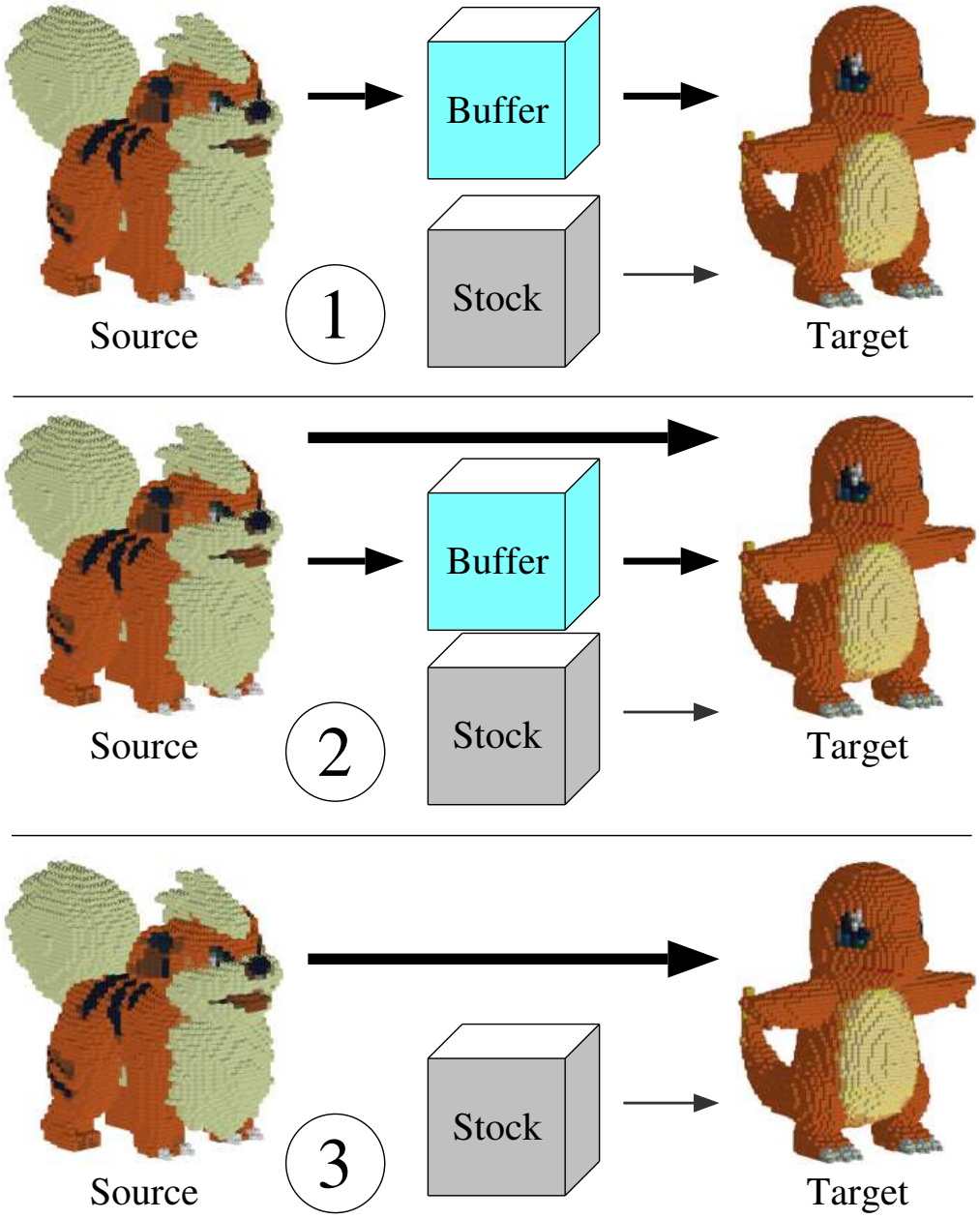

FIGURE 4.6: Three strategies under each approach. Strategy 1: Disassemble the source model completely to a buffer; then build the target from the buffer and a stock. Strategy 2: Build the target from the source, a buffer (disassembled bricks from the source), and a stock. Strategy 3: Build the target from the source and a stock.

another layer is disassembled to the buffer; so on and so forth till there is no brick on the source. If that happens, a stock will be used to provide a required brick for the target. Then, bricks are moved from the buffer and stock to the target, just as in Strategy

1. In other words, Strategy 2 tries to move bricks from the source to target directly; if stuck, it disassembles some bricks hoping that newly uncovered bricks can be used for transformation; but if no suitable one is found even when completely disassembling the source, the stock will be used.

In Strategy 3, no buffer is involved. Movable bricks are transported from the source to target; if no suitable brick is available on the top layer of the source, the stock will provide one. So on and so forth till the target model is finished.

4.3.2 Evaluation metrics

Coupling the strategies with the basic approaches, six algorithms are developed for shape transformation of LEGO models: FT1, FT2, FT3, RT1, RT2, and RT3. To evaluate the algorithms, three metrics are applied: cost (C), reuse rate (R), and overall performance (P). The cost is defined as the number of PnP needed to build the target model. Each PnP action is associated with a cost value of 1. This includes PnP actions on one LEGO brick from source to buffer, from source to target, from buffer to target, and from stock to target. Low cost suggests high efficiency.

The reuse rate is defined as

$$R = \frac{N_r}{N_s} \tag{4.5}$$

where N_r is the number of bricks on the target model that are originally from the source, and N_s is the number of bricks on the source model. N_r includes the bricks moved directly from the source and those moved indirectly through a buffer.

The overall performance is defined as

$$P = V \cdot \frac{R}{C} \tag{4.6}$$

where V is the number of voxels in the voxelized target model. P increases with the reuse rate and decreases with increasing cost. V puts the P metric on a relatively normalized scale against model size. It is not ideal to replace V with the number of LEGO bricks because different FT algorithms do not produce the same number of bricks on the target model, while V is same in all algorithms given a fixed model size.

4.4 Extensions

Besides the general pipeline, several extensions are also provided for users to customize the optimization of re-fabrication process.

Hollowing Hollowing is to remove the inner part of a model which is very important in some applications. It results in advantages such as economically using fewer bricks and reducing the total weight of the whole structure. Moreover, hollowing affects the stability of a structure, e.g. in Hong's work, hollowing is used to adjust a model's inner shape that the sculpture can stay balanced (Hong et al., 2016).

A simple hollowing approach is to remove the inner part of the shape by taking an input of number as shell size which is determined by users. The shell size is defined as the number of layers outside that remain solid.

Dangling Parts Removal In the process of removing bricks from the source model, it's common that dangling parts may appear in some intermediate states. For example, if one brick connects two parts of a LEGO sculpture, removing the brick will cut the whole sculpture into two parts, and if one of them is only connected with the removed brick, it will drop.

In our application, an algorithm based on graph theory is implemented. In graph theory, if two subgraphs that are connected to each other by only one node, the node is an articulation point. Applying it to each brick to be removed from the Source, we can determine whether it's an articulation brick, which can be tested by Depth First Search (DFS) algorithm. If it is an articulation brick, removing it will result in dangling parts, which may drop if there is no support. In this case, the dangling part will be moved to the Buffer.

Using Colors LEGO bricks are generally with different colors thus enable transformation between colorful models. Texture color is firstly converted into face color or vertex color first, and then the face color is rounded to the closest standard LEGO brick color in the voxelization process. There is a color chart of standard LEGO brick colors (*LEGO color chart*), and users can define their own standard color set based on the colors in

this chart. Next, during the merge algorithm (see above Section), another verification is added to allow merging of voxels: if both voxels are surface (visible) voxels and they have different colors, they cannot be merged. Inner voxels can be merged regardless, and they will be assigned a random color from the basic color list.

During the transformation process, the inner bricks in the target model will accept bricks regardless of colors. Otherwise, both the bricks type and color should be matched to allow transformation. Examples of LEGO models transformation using colors will be displayed in the following sections.

4.5 Results and Discussions

Eight triangle mesh models, as shown in Figure 4.7, were converted to LEGO models and used to test the proposed algorithms. Three types of tests were conducted. In the first two tests, performances of the algorithms were evaluated by the cost, reuse rate, and overall performance metrics without considering the color information. In the third test, the color information was incorporated, and a sequence of transformation was visually examined.

4.5.1 Test 1: Transform model A to other models

Model A in Figure 4.7 was transformed to the other seven models at a particular resolution which produced an average of 4500 voxels on each model. (A typical voxelization process cannot guarantee an exact number of voxels on a triangle mesh model but this does not affect the test.) LEGO models were generated using methods described in (Hong et al., 2016). Figure 4.8 shows the results of the algorithms under different performance metrics. The three strategies exhibit a descending order of cost as expected. Strategy 1 always disassembles the source model to a buffer, thereby having the highest cost. Strategy 3 does not use the buffer; hence, its cost is exactly the number of LEGO bricks on the target model. Comparing FT3 and RT3, we see that the number of bricks on the target generated by the two algorithms is different. In RT, the LEGO target was generated before transformation, and large bricks were used with priority. In FT, the LEGO target was generated on the fly, and available bricks from the source and buffer

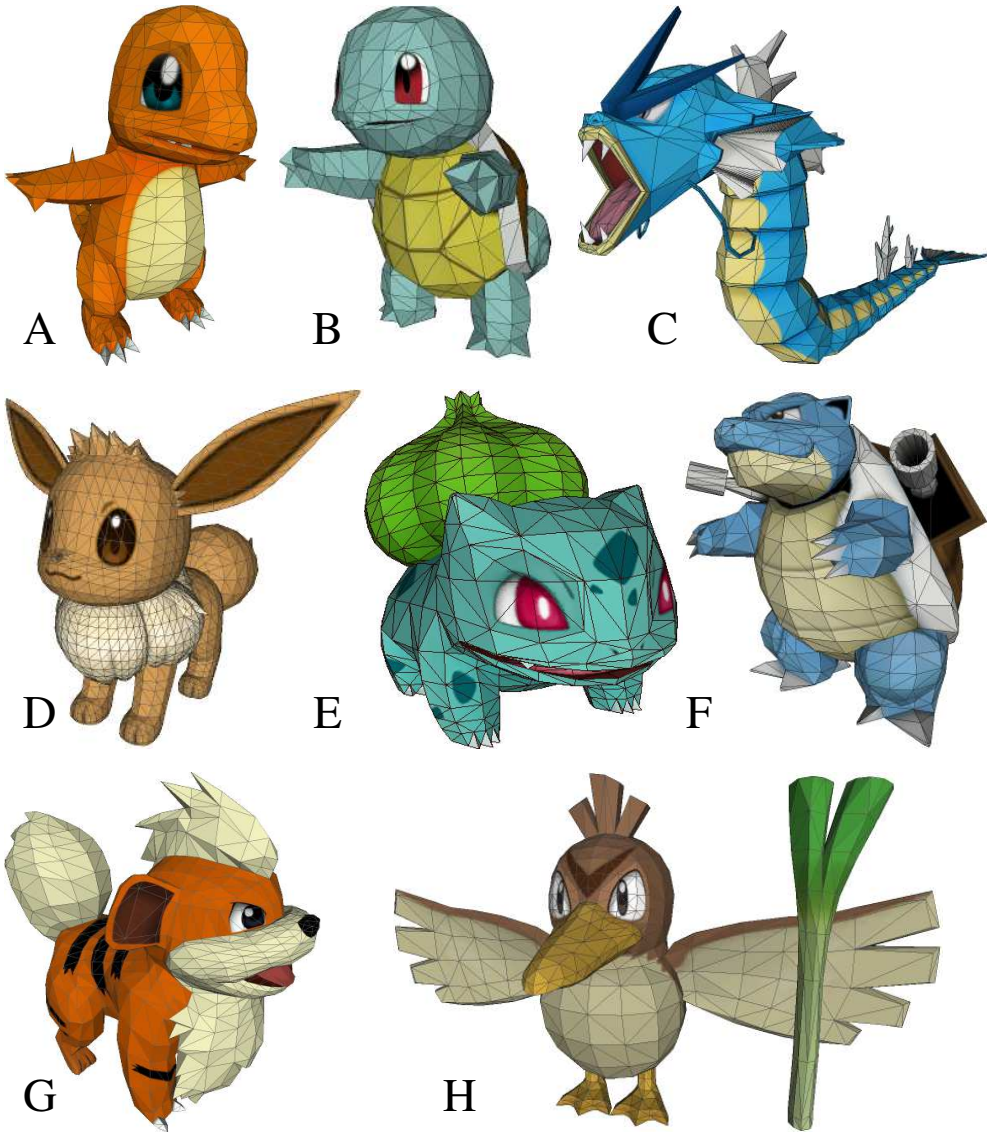

FIGURE 4.7: Triangle mesh models used to test the algorithms. The models were obtained from public space at ROEStudios. Pokémon is Copyright Gamefreak, Nintendo and The Pokémon Company 2001-2013"

were used before resorting to the stock; therefore, each FT algorithm produces different number of bricks on the target, and they use more bricks than the RT algorithms.

Intuitively, Strategy 1 and 3 should produce the highest and lowest reuse rate respectively. While this is indeed reflected in the trend of FT and RT respectively in Figure 4.8(b), we see an exception in model A-to-E transformation by FT. The reuse rate of FT2 is slightly higher than that of FT1. This is caused by the different number of bricks

in the target model generated by the different FT algorithms. FT1 has the source completely disassembled to the buffer; so many large bricks are available when the target is generated on the fly. FT2 exposes movable bricks layer by layer; hence, the available large bricks at any point of time are not that many; on-the-fly target generation is forced to use smaller bricks. Consequently, more bricks are used in FT2 than those in FT1, which makes the reuse rate of the former marginally higher than that of the latter. We also see that the reuse rate of a FT algorithm is higher than that of the corresponding RT algorithm. Being flexible in target model generation, FT naturally makes use of more bricks from the source than RT; however, these bricks tend to be smaller than the best-fit large bricks, resulting in weaker LEGO structures that have internal sliding planes. After all, FT increases the reuse rate by compromising the structural strength.

The overall performance, shown in Figure 4.8(c), measures efficiency of the algorithms. In the FT series, FT3 stands out as it achieves similar reuse rate as FT1 and FT2 at a much lower cost. This suggests that for on-the-fly target generation, it is not efficient to involve a buffer because a brick is very likely to be reused given only seven types of bricks without considering color (Figure 4.2). In the RT series, RT2 is the best except for model A-to-C transformation. This suggests that if the target bricks are fixed types, making moderate use of a buffer produces a good balance between the cost and reuse rate.

4.5.2 Test 2: Transform eight models to each other

This test was conducted at seven resolutions, i.e. seven average model sizes, from 500 to 4500 voxels per model. The eight models were transformed to each other and there were $7 \times 8 = 56$ transformations at each model size. Figure 4.9 shows the average results over the 56 transformations. It is obvious from Figure 4.9(a) that the cost increases linearly with the model size. Strategies 1, 2, and 3 have a consistent descending cost at a particular model size. The average of these transformations reveals the inherent cost associated with each strategy. A FT algorithm has a higher cost than its corresponding RT algorithm because FT produces a target model made of more bricks.

Figure 4.9(b) shows that Strategies 1 and 2 produce very similar reuse rate, suggesting that completely disassembling the source upfront (Strategy 1) is not worthwhile;

it is better to do partial disassembly to save some cost while maintaining a high reuse rate. When Strategy 3 is coupled with the FT approach (i.e. FT3), the reuse rate is actually quite high, comparable to the best of RT algorithms; however, the reuse rate of RT3 is significantly lower than that of RT1 and RT2. This implies that when the target model is prior generated, disabling the buffer makes it difficult to find matching bricks from the source, and on average more than half of the target bricks would come from the stock. It is also seen from the figure that the reuse rate is not that sensitive to the change of model size above 1700 voxels per model. Smaller model size (fewer than 1700 voxels) does reduce the reuse rate because the reduced brick samples in smaller models make it harder to find matching bricks.

In Figure 4.9(c), statistics of the overall performance shows that FT3 is much better than other algorithms. Within RT algorithms, RT2 is statistically the best. Note that the overall performance is only an empirical metric. Different scenarios may emphasize different evaluation criteria. It may be possible that in certain scenario the cost metric must be redefined, and the relative performance of the algorithms will vary from our results; nevertheless, the contribution of this study is in framing the LEGO transformation problem as a multi-objective optimization process, in which different strategies should be explored to achieve some balance between objectives.

4.5.3 Test 3: Transformation with color information

This test incorporated the color information of each triangle mesh model when building the corresponding LEGO model. A legitimate match between a source and a target brick should be the same color as well as the same shape. Due to the fact that more than one hundred colors were used, finding a match was much less likely than in Tests 1 and 2 where the color information was ignored; however, in our implementation of the algorithms, an interior brick can take on any color; only exterior bricks must find exact color match. Consequently, although the average reuse rate in this test is lower than that in Tests 1 and 2, it is not down to zero.

Figure 4.11 shows typical FT sequences between two models. FT1 first disassembles the source to a buffer, displayed next to the stock, at a cost of 2755 PnP. Then it builds the target using bricks from the buffer and stock. Interior bricks of the target model can

be of any color, while exterior bricks must be the surface color. FT2 moves the source bricks to the buffer only if there is no suitable ones. The bottom of the target (feet of the Farfetch'ed) is made of yellow bricks, and they are hard to find from the source; hence, more than half of the source model are disassembled before the feet can be built. The cost and reuse rate of FT2 are slightly higher than those of FT1 for reasons discussed in the previous sections: the target built by FT2 has more bricks than that by FT1. FT3 does not use the buffer. The source is only partially disassembled when the target is fully built. FT3 produces the lowest cost and reuse rate.

Figure 4.12 shows typical RT sequences between another two models. As both models contain orange surface bricks, the reuse rates of RT1 and RT2 are relatively high, above 60%. In RT3, the interior of the target contains many black bricks, which are used to indicate bricks obtained from the stock and placed on the interior of the target. (They do not have to be black; any color is fine.) Comparing all six sequences in Figures 4.11 and 4.12, only RT3 shows large quantity of black bricks. This means that the other algorithms, including FT3, are able to find suitable interior target bricks from the source as it is only the shape of the bricks that must match. Though FT3 does not use the buffer just like RT3, it has the liberty to build the target based on available bricks; therefore, it does not use the stock to provide many interior target bricks and its reuse rate is much higher.

4.6 Conclusion and Future Work

Shape transformation of LEGO models has been investigated as a multi-objective optimization problem. Two basic approaches have been proposed. The first approach, FT, allows for on-the-fly generation of a target model based on available LEGO bricks from a source model. The second approach, RT, requires both the source and target models to be generated before transformation. Three strategies are adopted under each approach, which leads to six algorithms. A cost, a reuse rate, and an overall performance metrics have been applied to evaluate the algorithms. Strategy 1 produces the highest cost and reuse rate on average. Strategy 3 produces the lowest lost and reuse rate. Strategy 2 achieves a balance between Strategies 1 and 3. It gives the best overall performance in

RT algorithms. In FT, owing to on-the-fly target generation, Strategy 3 obtains a reasonably high reuse rate at a low cost; hence, it outperforms the other strategies in overall performance.

FT achieves better results than RT due to flexible model generation; however, this has the disadvantage of producing relatively weak LEGO structures, which have internal sliding planes. FT also tends to produce a model made of more bricks than RT.

In this study, we only used one brick as a unit of PnP. In practice, sometimes a group of LEGO bricks can be reused as a whole. Future work along this direction may look into novel LEGO model generation methods that tend to produce modularized LEGO models. Then, shape transformation may be achieved at a modular level, not at a single brick level. Successful implementation of such modular-oriented methods would lead to highly efficient LEGO transformation algorithms.

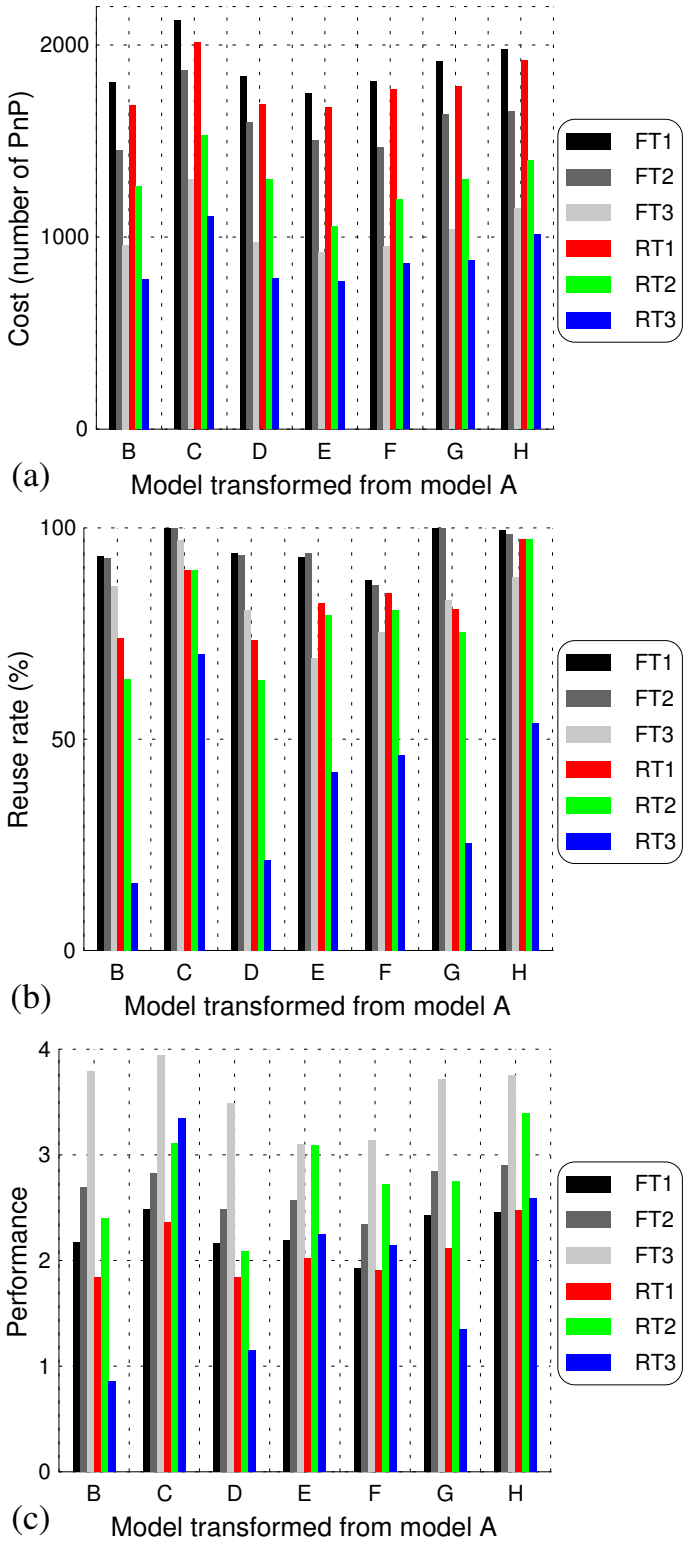

FIGURE 4.8: Performances of the algorithms in transforming model A to other models. (a) Cost, (b) reuse rate, and (c) overall performance of transformation.

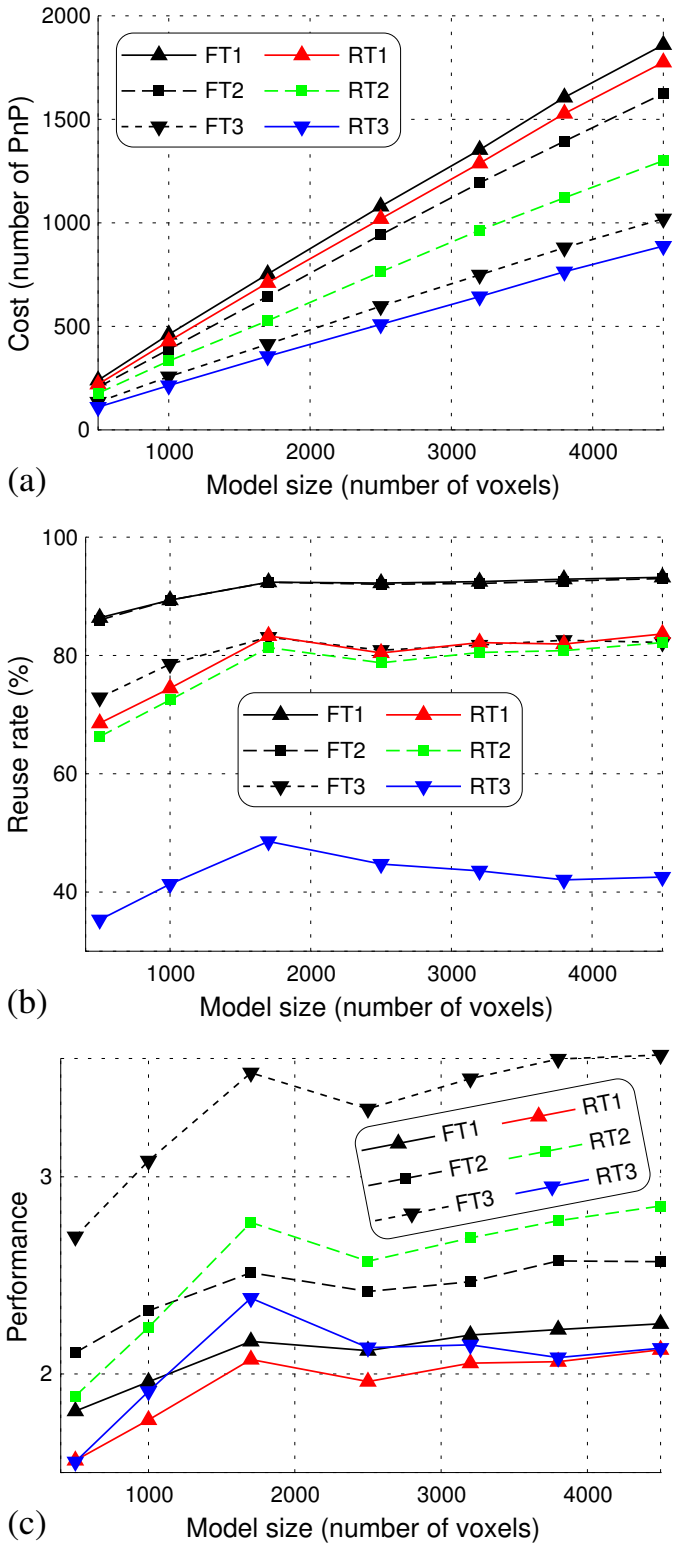

FIGURE 4.9: Average performances of the algorithms in transforming eight models to each other. (a) Cost, (b) reuse rate, and (c) overall performance of transformation.

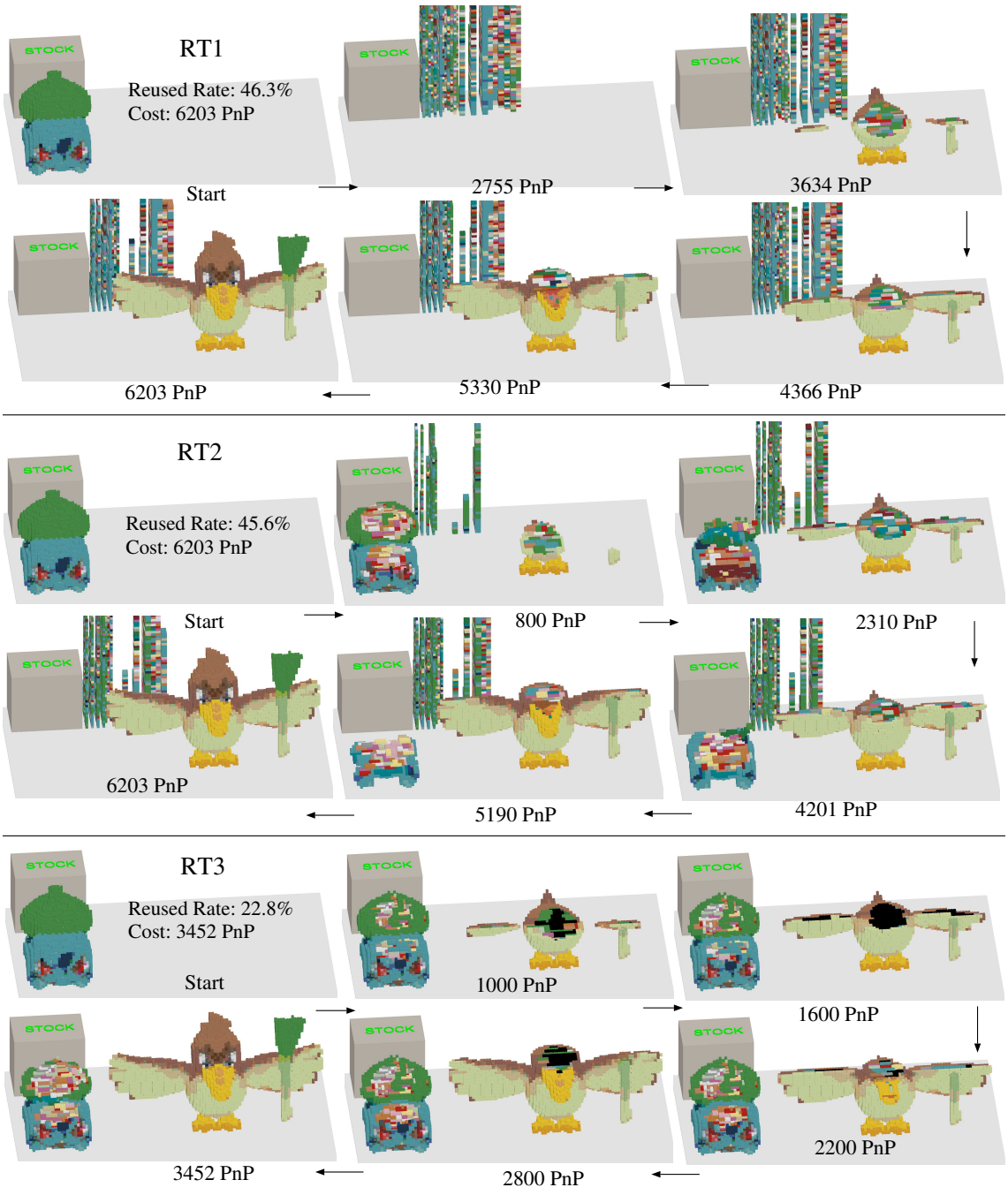

FIGURE 4.10: Typical RT sequences between two models: tranformation from Bulbasaur to Farfetch'd.

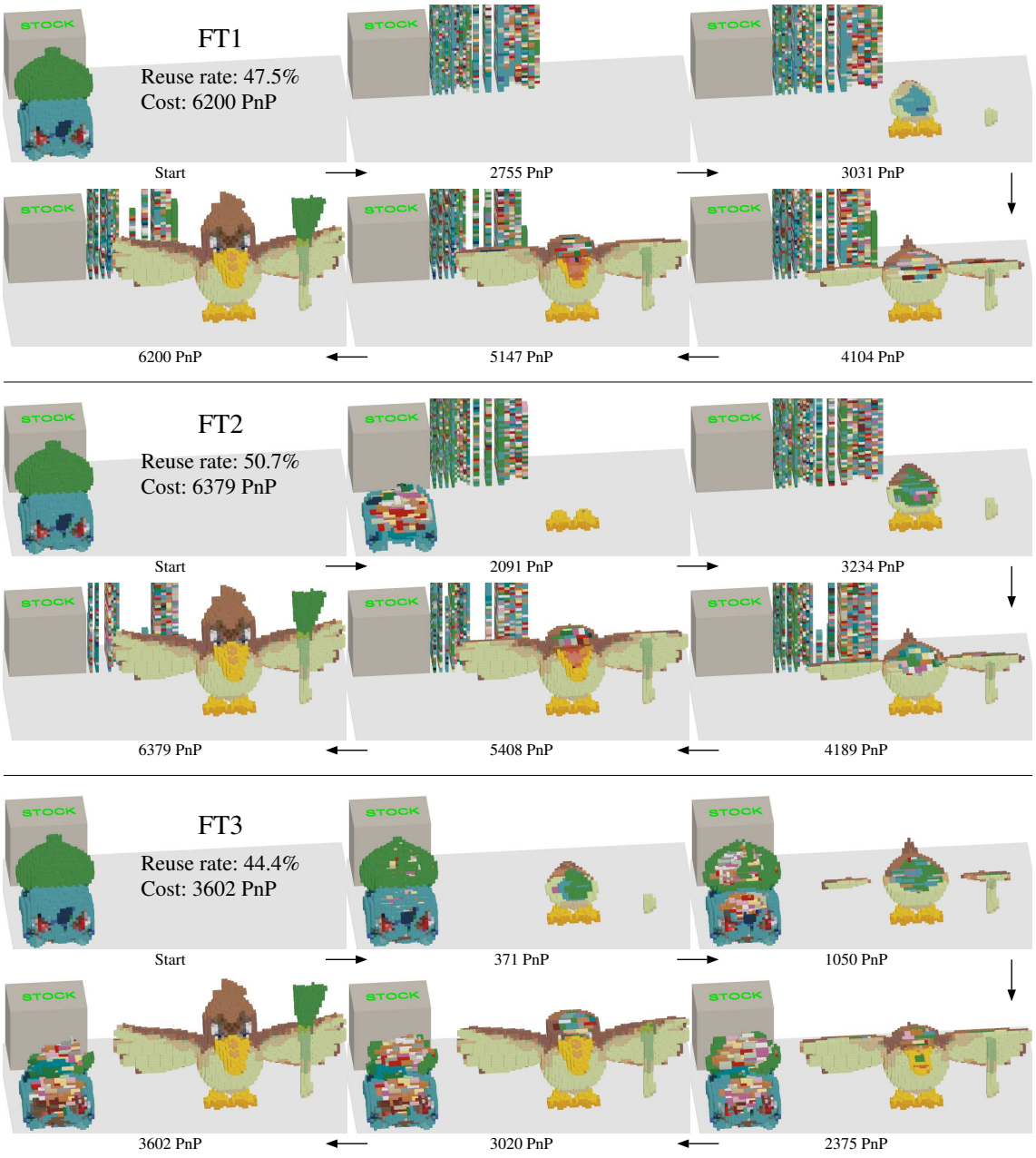

FIGURE 4.11: Typical FT sequences between two models: tranformation from Bulbasaur to Farfetch'd.

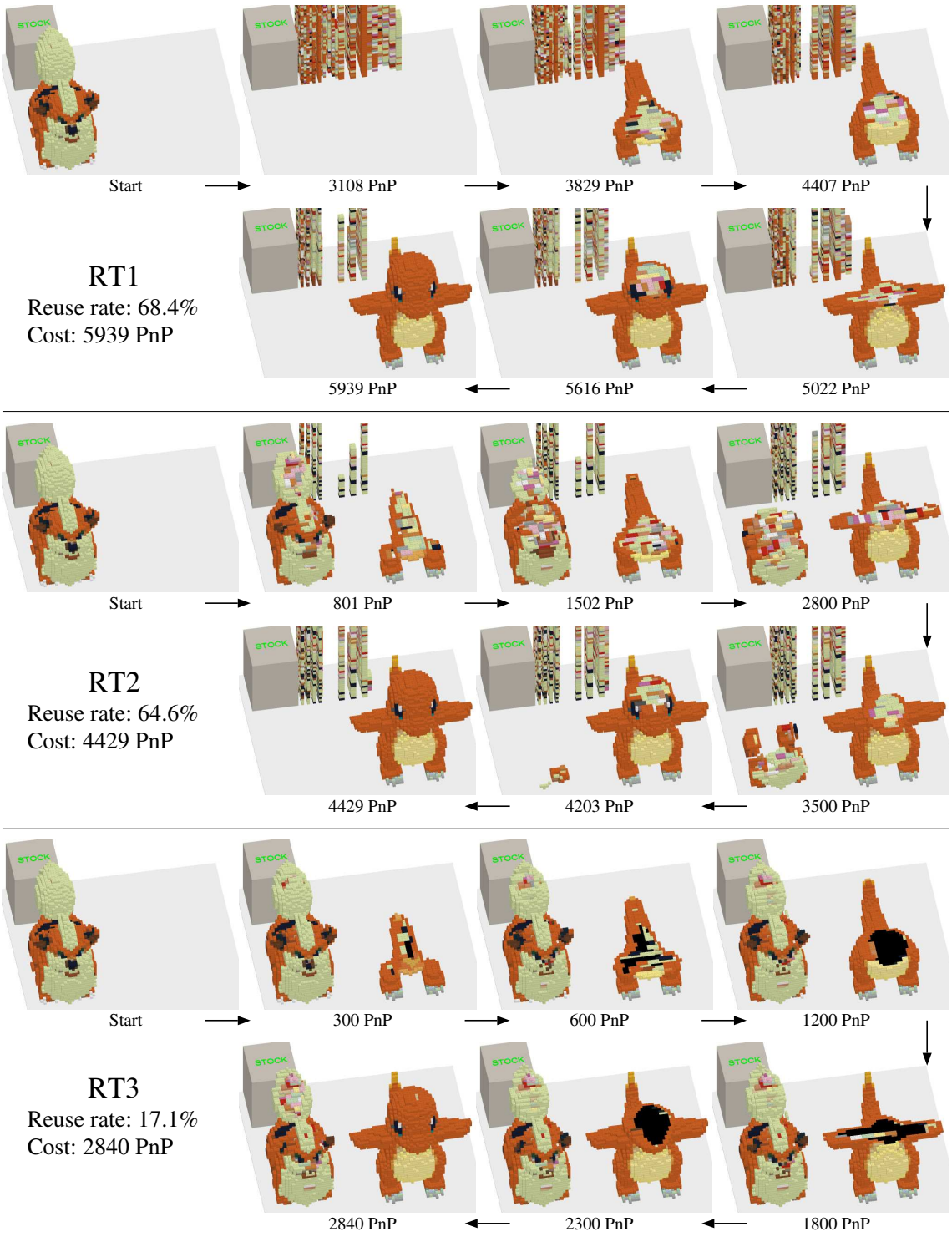

FIGURE 4.12: Typical RT sequences between two models: transformation from Growlithe to Charmander.

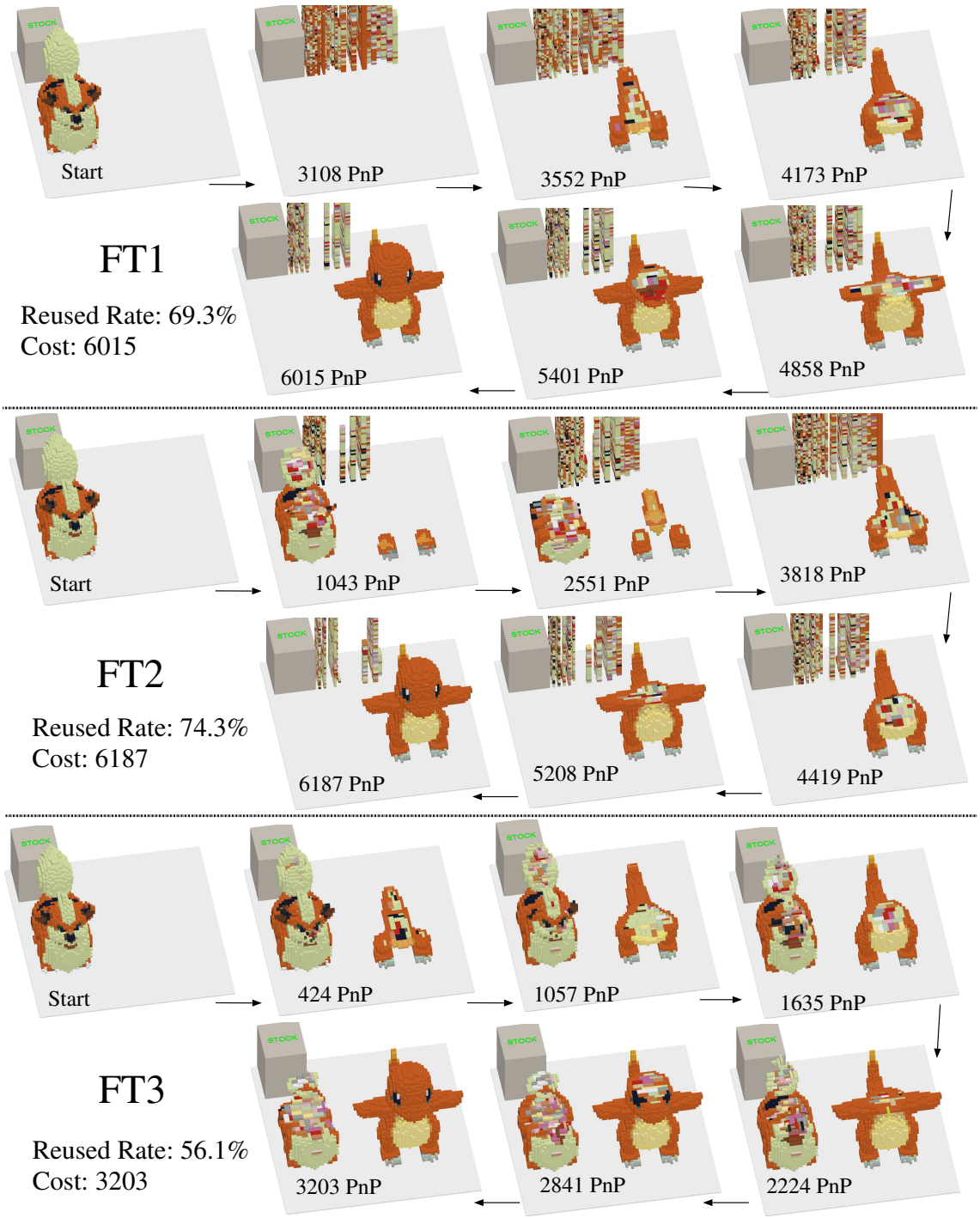

FIGURE 4.13: Typical FT sequences between two models: transformation from Growlithe to Charmander.

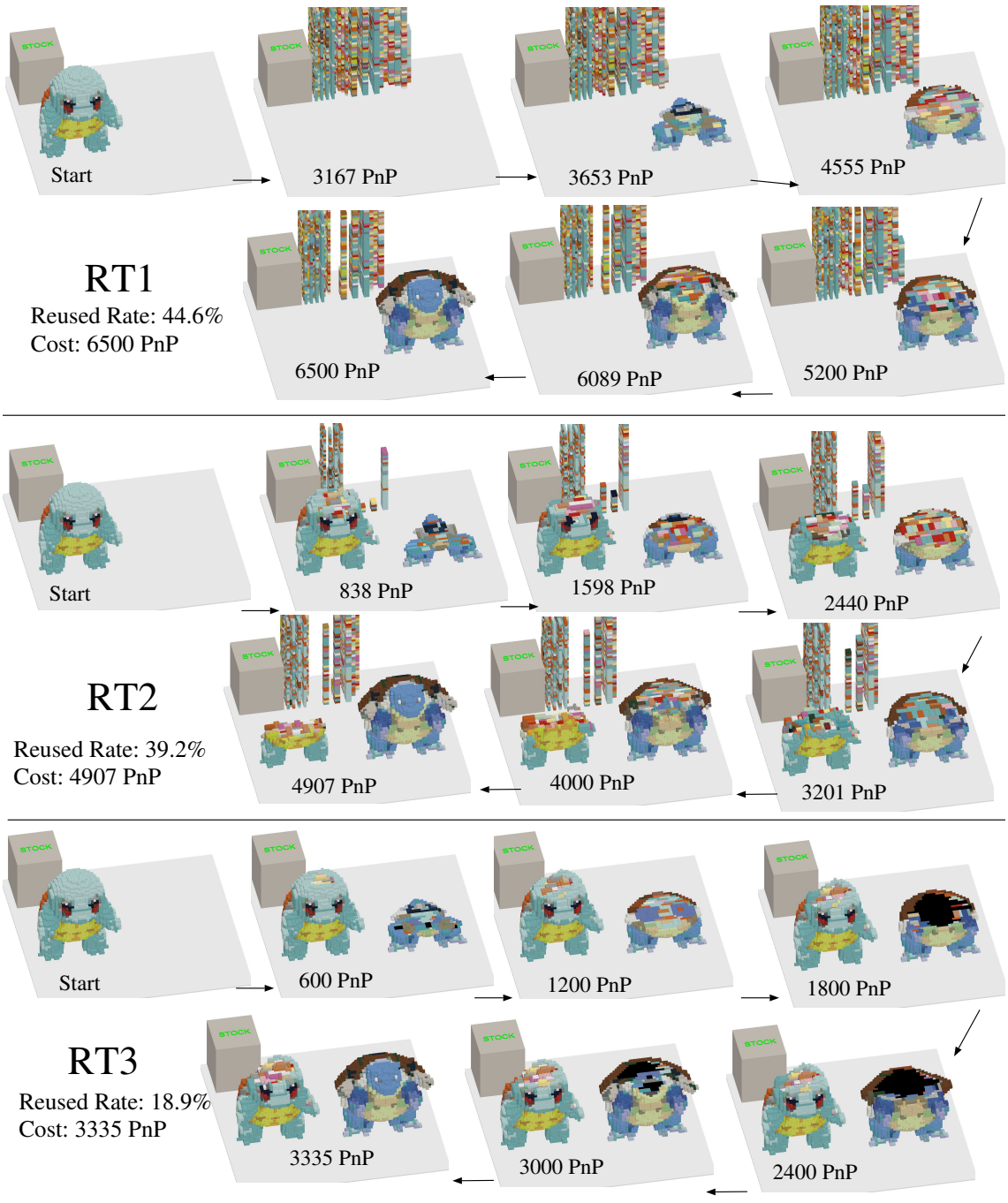

FIGURE 4.14: Typical RT sequences between two models: transformation from Squirtle to Blastoise.

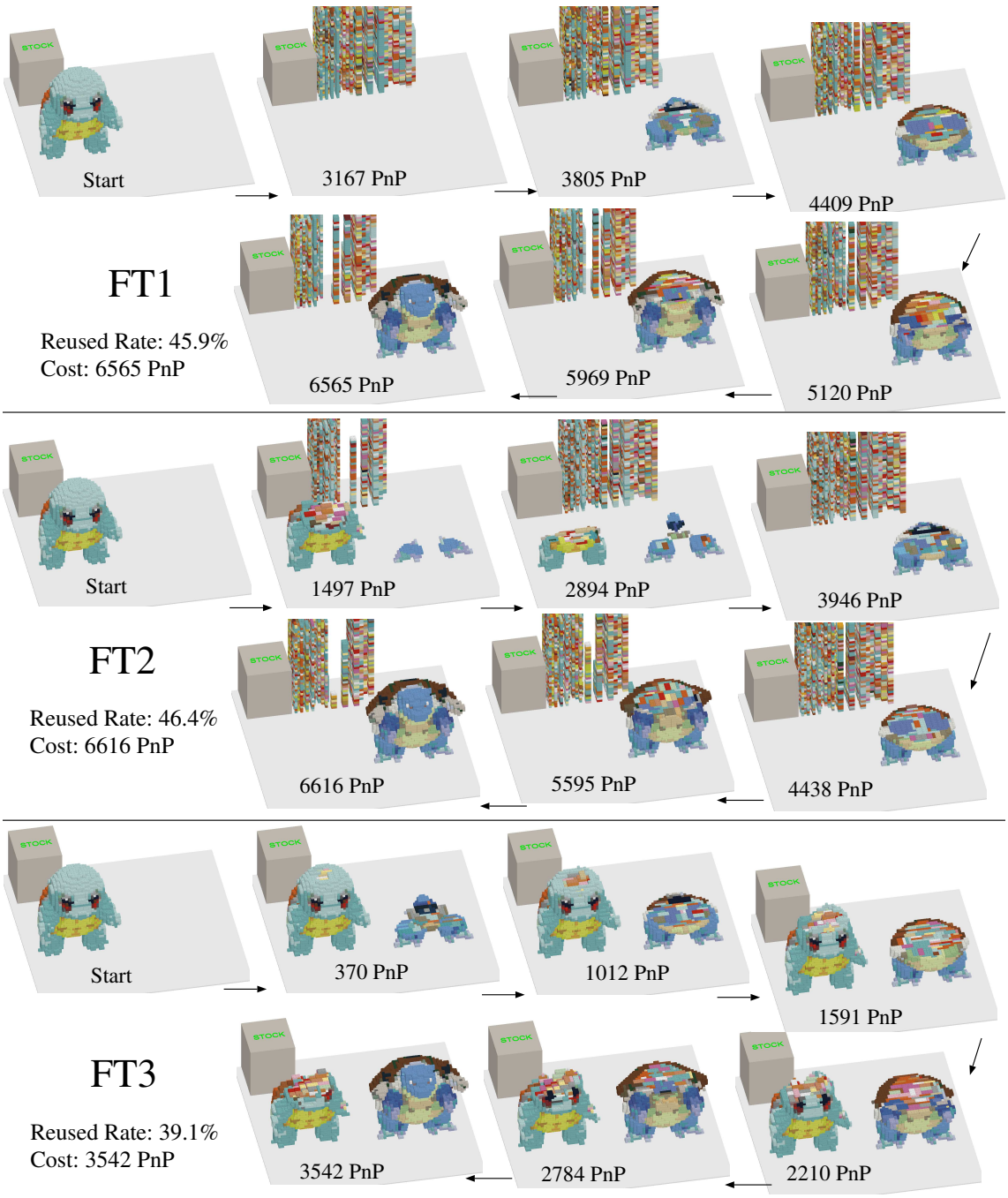

FIGURE 4.15: Typical FT sequences between two models: transformation from Squirtle to Blastoise.

Chapter 5

Sequencing for Chain Structures

"I do not think there is any thrill that can go through the human heart like that felt by the inventor as he sees some creation of the brain unfolding to success."

— Nikola Tesla

5.1 Overview

In this chapter, the folding sequencing in the shape transformation of rasterized chain structures is investigated. Different from traditional origami, which is to fold 2D shape (e.g. paper sheet) into 3D structure, our application is aiming at applications in folding rasterized 1D chain into rasterized 2D or 3D shape. Specifically, the sequencing of the transformation from one shape into another based on algorithms such as Hamiltonian paths and spanning trees, which have been demonstrated that any 2D or 3D geometry can be described by a 1D chain (Griffith and Jacobson, 2004; Bachrach, Zykov, and Griffith, 2009). Figure 5.2 and 5.3 show example 2D and 3D space filling primitive and the connected linear strings.

The proposed folding sequence planning has two processing stages:
(1) Preprocessing: Rasterization and Hamiltonian path searching on lattice,
(2) Folding sequencing, as illustrated in Figure 5.1.

The method accepts two continuous images (2D) or closed polygonal meshes (3D) as inputs, which are rasterized into voxels and consequently structured as a form of 1D chain. Then, the sequence to fold one to another is generated.

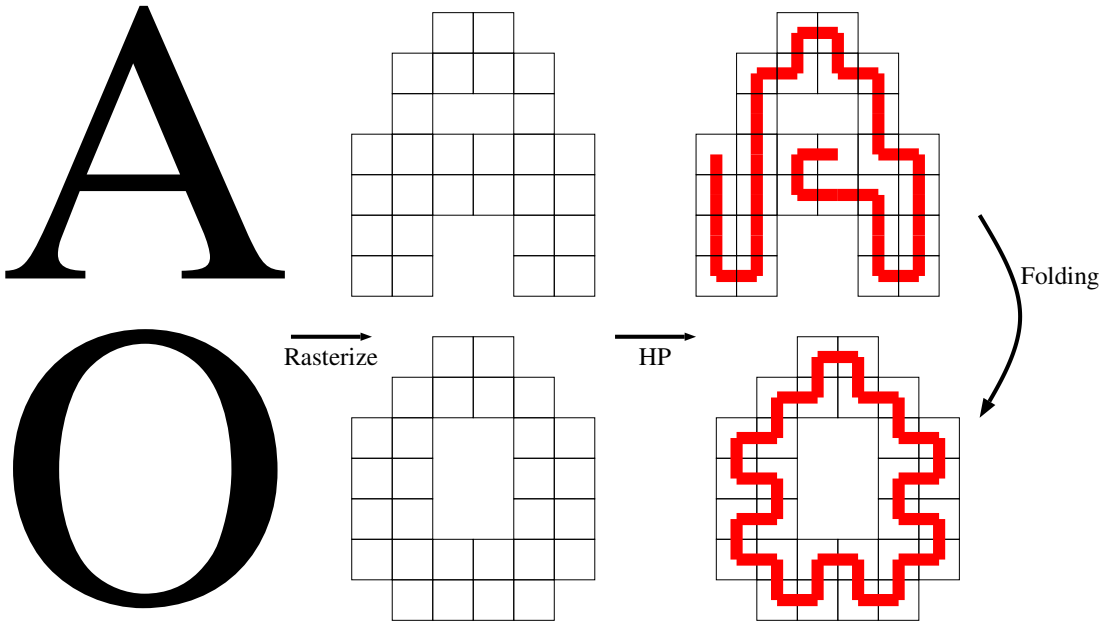

FIGURE 5.1: Overview of folding sequencing. Letter A and O are rasterized into 28 pixels form, and a Hamiltonian Path (HP) problem is solved to transform them into strings. The last step is to fold the A string into O string.

5.2 Preprocessing

5.2.1 1D Chain with Fixed Length

Rasterization of 2D/3D Structures with Controllable Scale

As described in Chapter 3 and 4. Rasterization of a polygonal mesh is a well studied topic. In most case, the length of the 1D chain which is folded into 3D shape are fixed and the scale of the 3D shape needs to be controlled. [1]

Given a 1D chain with length L (The number of voxels). The volume V of the polygonal mesh is computed with the method described by Zhang et al. (Zhang and Chen, 2001). Assuming the side length of each voxel is l (Equation 5.1), the value of l can be easily computed by Equation 5.2.

$$l^3 \times L = V \tag{5.1}$$

[1] Three-dimension (3D) in most cases includes two-dimension (2D). To clear ambiguity, in the following part the 2D and 3D cases are not differentiated.

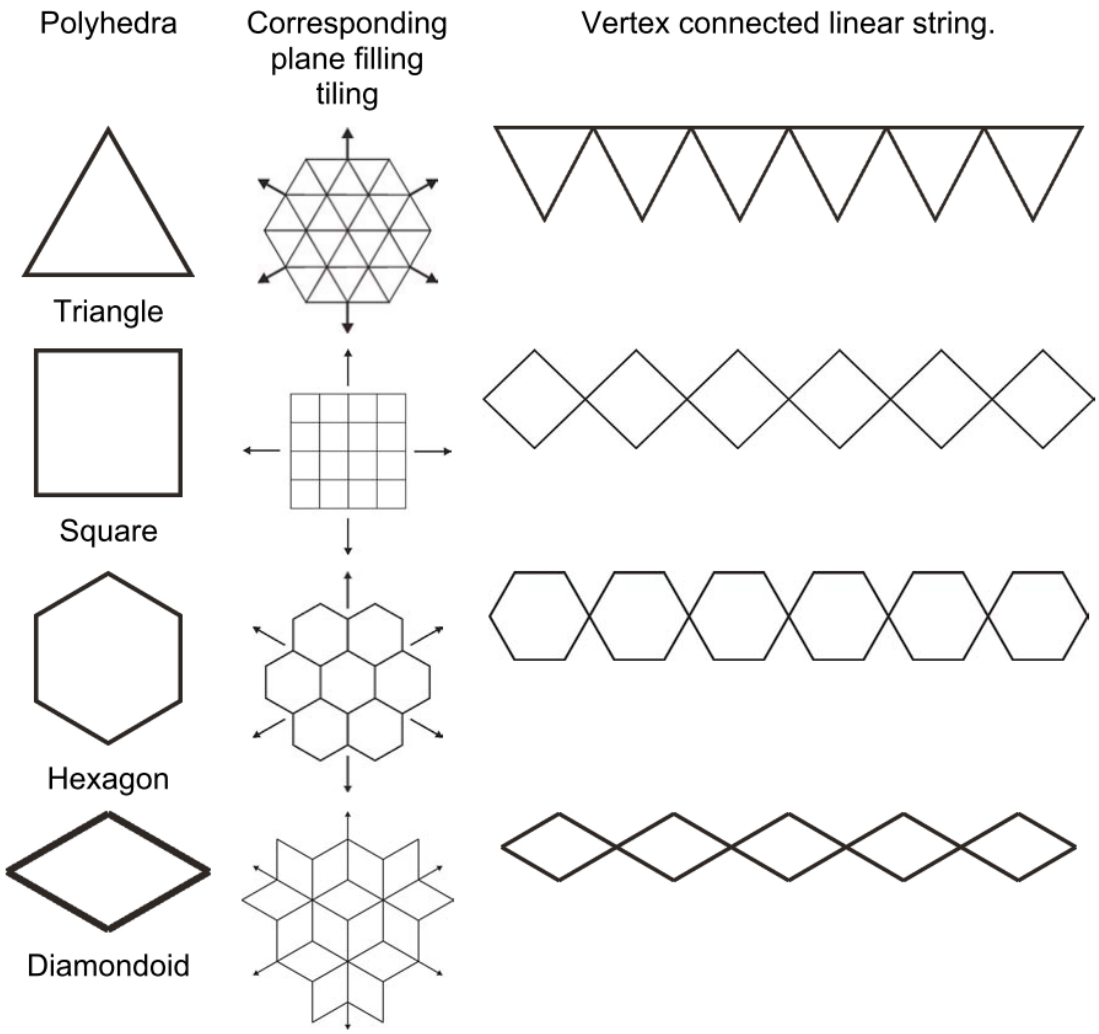

FIGURE 5.2: Polygonal primitives, their plane tilings, and schematic of their vertice connected strings (Griffith and Jacobson, 2004).

$$l = (V/L)^{1/3} \tag{5.2}$$

The side length of voxel is used to control the scale of rasterization. However, due to the discrete feature of raterization, the number of voxels of the rasterized model is expected to be different from L. Then the rasterized shape is modified in the next step so that the length of it can be controlled to be L.

Prune and Append

The Rasterized model can be regarded as a graph with each voxel being considered as node, and an edge represented as the connection with its neighbors (Figure 5.4. Assuming that a vertex is odd vertex if its sum of x,y,z coordinates is odd, otherwise it's even vertex). This graph has following properties:

1. each vertex has degree 1 - 6;

2. The boundary vertices (corresponding to the boundary voxels) have degree 1 - 5, and

3. The inner vertices always have degrees 6 (6 neighbors);

Let's consider the problem of folding this 3D model from 1D chain. Reversibly, to fold a 1D chain into 3D lattice shape can be regarded as the problem to find the Hamiltonian Path (HP) in the above mentioned graph. The HP problem in a general grid/lattice graph is shown to be NP-complete, but there are studies on the necessary and sufficient conditions of this problem by considering it as a bipartite problem (Itai, Papadimitriou, and Szwarcfiter, 1982) and algorithm exists to provide a linear time solution (Dong Chen, Shen, and Topor, 2002). The conditions are listed as follows, and readers may refer to Appendix A for further information.

1. If the starting vertex s and the terminal vertex t share an edge in the graph, it's a Hamiltonian circuit (HC) problem, and no vertex of degree 1 should exist; otherwise, it's a HP problem, no more than 2 vertices of degree 1 should exist.

2. If a vertex has degree 2, then both edges incident to it must be in the HP (HC).

3. The graph G must be compatible if (1) The number of even vertices equals the number of odd vertices ($|V_{odd}| = |V_{even}|$, G is even), and s and t have different color

or (2) G is not even, e.g. the number of odd vertex is greater than the number of even vertex ($|V_{odd}| = |V_{even}| + 1$), and s and t are colored by the majority color ($s, t \in V_{odd}$).

By following the above requirements, the graph generated from the 3D model should be pruned or appended so that an HP algorithm described in Appendix A can be applied to find a HP in the Graph. There are a few rules to follow in this step:

1. Keeping two vertices of degree one at most as the start and terminal vertex;
2. Balancing the Graph (Equalize the number of even and odd vertices);

5.2.2 1D Chain with Arbitrary Length

Griffith et al. described a spanning tree method to solve the problem of folding 1D into 2D or 3D (Griffith and Jacobson, 2004). The method is to divide each voxel by 2 in each direction, therefore, each voxel is divided into 4 sub-voxels in 2D plane and 8 sub-voxels in 3D space (Figure 5.5).

The 3D model is first rasterized with half of the normal resolutions (with two times of the usual edge length). Then each voxel in divided into 4 (2D) or 8 (3D) equal sub-voxels, and it's already a Hamiltonian path as the perimeter of the original graph.

5.3 Folding Sequencing

The motion sequence proposed in the work of Ding et al. is applied here (Ding and Lu, 2013). Nevertheless, the forward and inverse kinematics calculation of the specific discrete motion are ignored here for two reasons:

(1) it's beyond the range of this thesis;

(2) The 1D chain is an abstract concept, rather than a speculative example which is hard to do computation.

5.3.1 Adjacent Geometry

The configuration of a 1D chain can be expressed by a sequence of joint angles, which described the relative position between adjacent voxels. For example, in Figure 5.6, the five voxels chain has three statuses. Initially, the angle sequence is $\alpha_1 = [0, 0, 0, 0]$. In the second status, it becomes $\alpha_2 = [90, -90, 90, -90]$ and in the third status, it's $\alpha_3 = [-90, -90, 90, 90]$.

5.3.2 Motion Sequence

Due to the discrete feature of the folding modules, the ordinal of modular movement is defined as independent variable in this sequence, which contains the following three aspects: the angle of joint motion, the number of joint and the ordinal of joints. A motion sequence matrix Seq is defined to describe them together,

$$Seq = \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1j} & \dots & S_{1m} \\ S_{21} & S_{22} & \dots & S_{2j} & \dots & S_{2m} \\ \vdots & \vdots & \ddots & \cdots & & \vdots \\ S_{i1} & S_{i2} & \dots & S_{ij} & \dots & S_{im} \\ \vdots & \vdots & & \cdots & \ddots & \vdots \\ S_{n1} & S_{n2} & \dots & S_{nj} & \dots & S_{nm} \end{bmatrix} \tag{5.3}$$

where n rows mean the number of the module during the movement, m columns represent the ordinal of the module, and S_{ij} indicates the rotating angle of the jth module at the ith step.

When the folding sequence $\theta = [\theta_1, \theta_2, \dots, \theta_m]$ and the ordinal of moving $a = [a_1, a_2, \dots, a_n]$ are known, The motion sequence can be obtained by using Equation 5.4,

$$Seq(i, j) = \begin{cases} \theta_j, & i = a_j \\ 0, & i \neq a_j \end{cases} . \tag{5.4}$$

Take the system with five voxels as an example, shown in Figure 5.6. In the first transformation, the folding sequence is $\theta = [90°, -90°, 90°, -90°]$, and the ordinal of

voxel motion is $a = [3, 2, 4, 1]$. The motion sequence is

$$Seq_1 = \begin{bmatrix} 0 & 0 & 90^\circ & 0 \\ 0 & -90^\circ & 0 & 0 \\ 0 & 0 & 0 & -90^\circ \\ 90^\circ & 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

Similarly, the second folding transformation can be expressed by the motion sequence as

$$Seq_2 = \begin{bmatrix} 0 & 0 & 0 & 180^\circ \\ -180^\circ & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.6}$$

5.3.3 Feasible Folding Sequence

Before introducing the method to generate a feasible folding sequence, two operations are introduced first.

Isomorphism Judging Since the folding materials are consist of identical voxels, the homogeneous feature of voxels may result in a fact that two folding results are the same although the folding sequences of them are different (Figure 5.7).

For simple structure, an easy step can be made by directly comparing the adjacent geometry, i.e. the angle sequence of the two configuration: $\alpha_x = [\alpha_{x1}, \alpha_{x2}, \ldots, \alpha_{xn}], \alpha_y = [\alpha_{y1}, \alpha_{y2}, \ldots, \alpha_{yn}]$. if $\alpha_{xi} = \alpha_{yi}$ or $\alpha_{xi} = \alpha_{n+1-yi}, i = 1, 2, \ldots, n$ then the two configuration is equivalent.

Another method based on eigen value and eigen vector analysis of adjacency matrix proposed by Chang et al. (Chang et al., 2002) can identify isomorphism which exists due to the special shape of modules. The adjacency matrix of an n voxels configuration is defined as,

$$D_i = \begin{bmatrix} 0 & d_{12} & \dots & d_{1n} \\ d_{21} & 0 & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & 0 \end{bmatrix} \tag{5.7}$$

where D_i indicates the adjacency matrix of the ith configuration, and entry d_{ij} is the distance between center of the ith voxel and that of the jth voxel. Assume that the eigenvalues of the above matrix are calculated and sorted as $\lambda_i = [\lambda_{i1}, \lambda_{i2}, \dots, \lambda_{in}]$ with corresponding eigenvectors $X_i = [X_{i1}, X_{i2}, \dots, X_{in}]$,

It has been proved that if

$$\begin{cases} \lambda_{ai} = \lambda_{bj} \\ X_{ai} = X_{bj} \end{cases} \quad i = j = 1, 2, \dots \tag{5.8}$$

then the two configurations a, b are isomorphic.

Collision Detection Collision detection is widely applied in computer graphics, Computer-Aided-Design (CAD), Computer-Aided-Manufacturing (CAM), etc. It's highly dependent on the geometry with a certain volume, e.g. the cuboid components are different from the triangle prism components. There are algorithms based on bounding box, distance calculation, Voronoi diagram and so on (Ding, Mannan, and Poo, 2004; Jiménez and Segura, 2008; Xie, Yang, and Zhu, 2007).

5.3.4 Folding

The problem of folding sequencing is processed as follows:

1. Get the adjacent geometry, i.e. angle sequences of the initial status and the target status, α_i and α_t. Then the folding sequence is $\theta = \alpha_t - \alpha_i$.

2. From the folding sequence, one can get the folding voxels $v = [v_1, v_2, \dots, v_m]$ which satisfying $\alpha_v \neq 0$.

3. Construct a folding tree (Figure 5.8) with the folding voxels in the last step, which proposed in the work of Ding et al. (Ding and Lu, 2013). Recursively use the method described in Section 5.3.3 to select one possible configuration.

4. Build the sequencing matrix by the route obtained from last step.

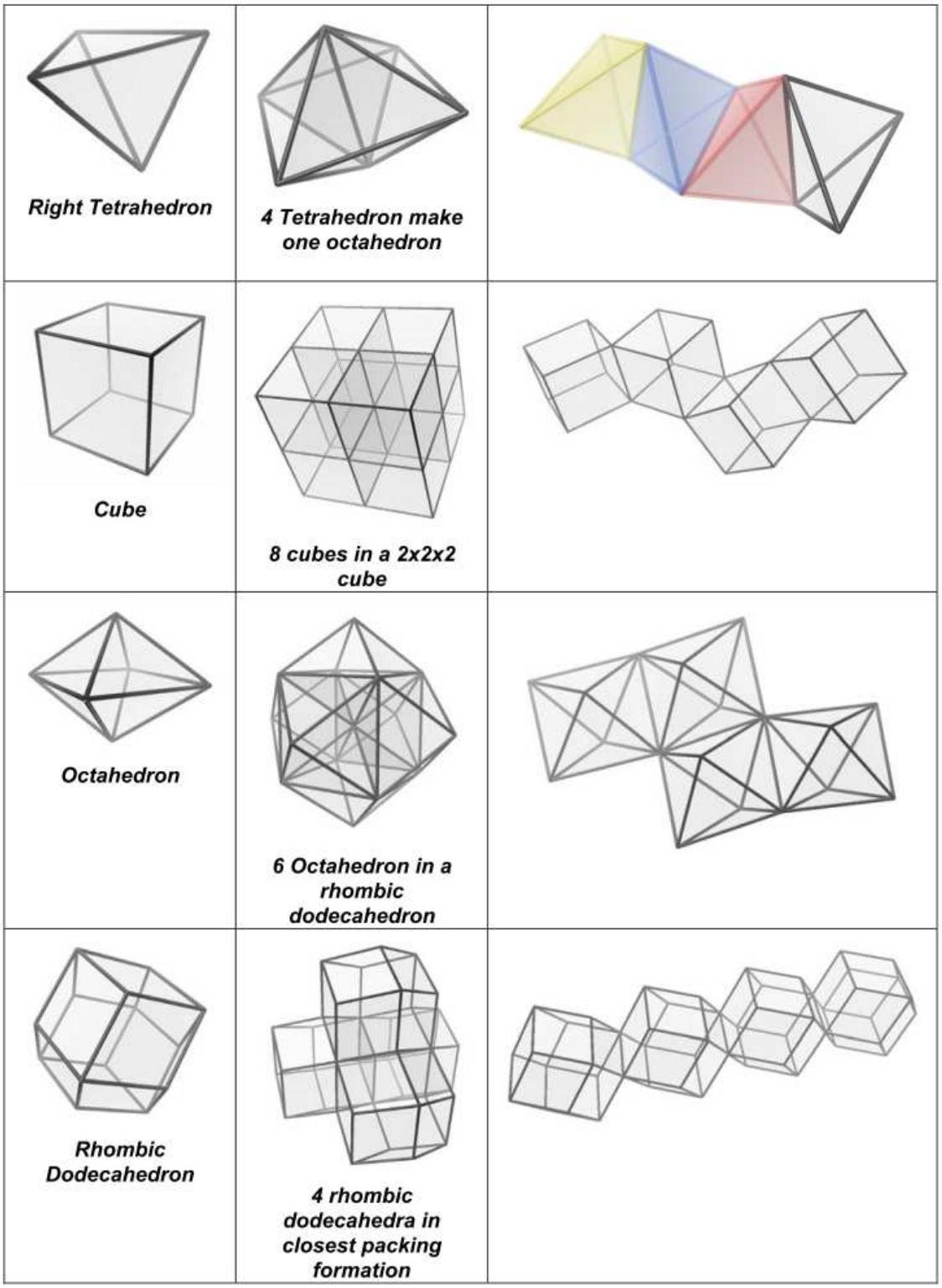

FIGURE 5.3: Space filling polyhedra, their crystal stacking, and representation as strings of edge-connected polyhedra as might be useful in folding (Griffith and Jacobson, 2004).

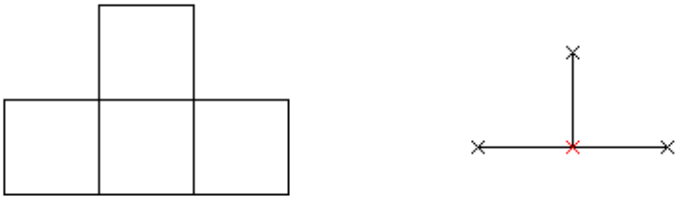

FIGURE 5.4: A graph constructed from a 2D models of 4 voxels. The red vertex is even vertex while the black vertices are odd vertices.

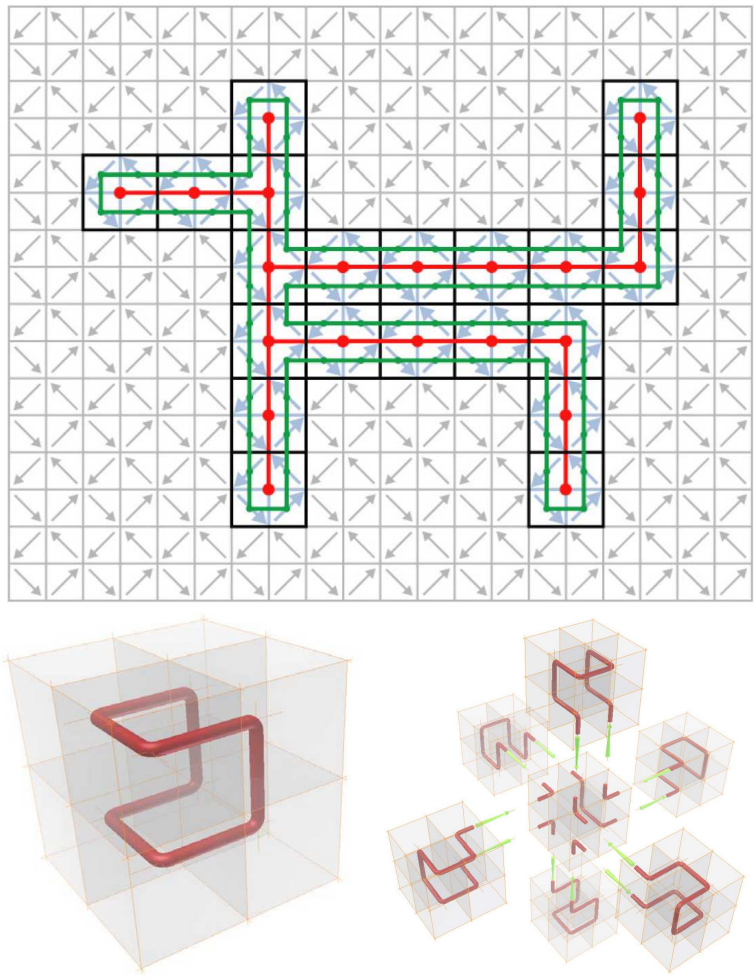

FIGURE 5.5: Voxel division method (Griffith and Jacobson, 2004). Above: a 2D dog like graph (black) with each square divided into 4 sub-tiles(grey). An Hamiltonian Circle (green) that follows the perimeter of the original graph (red); Below: The case for 3D shape.

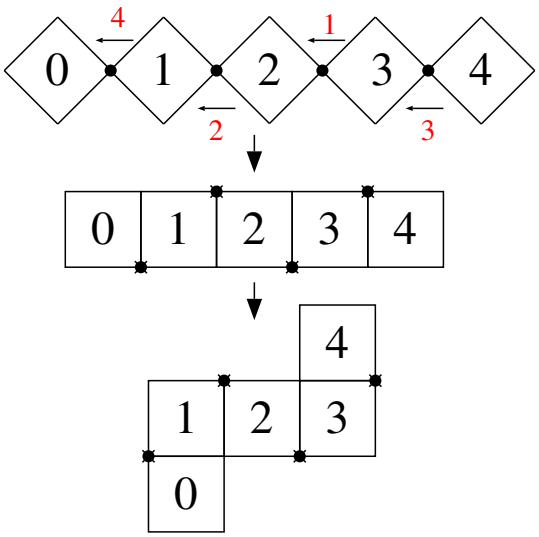

FIGURE 5.6: A simple example consists of 5 cubes demonstrate the motion sequence. The point is the joint between two adjacent voxels.

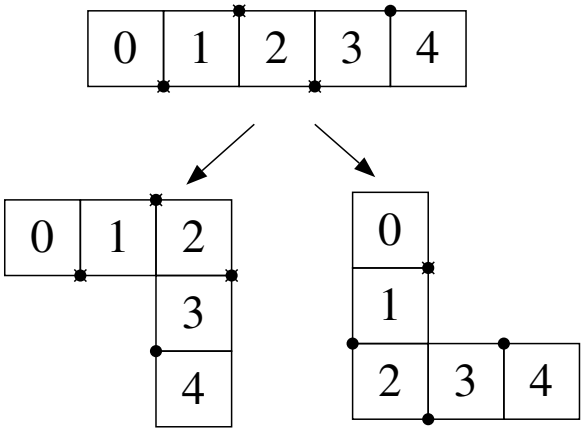

FIGURE 5.7: Equivalent modular motion sequence.

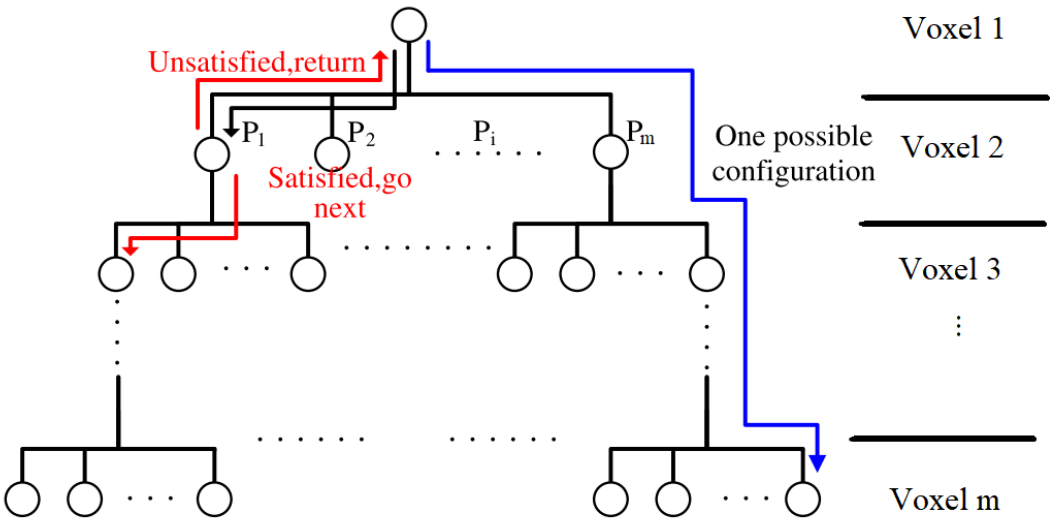

FIGURE 5.8: The configuration tree.

Chapter 6

Conclusion and Future Work

"The significant problems we have cannot be solved at the same level of thinking with which we created them."

— Albert Einstein

6.1 Conclusion

This thesis investigated the sequencing problem of shape transformation of rasterized 3D structures by three cases studies: the problem of pick and place operation for homogeneous components (voxels), heterogeneous components (bricks) and folding operation for voxels.

By developing a cost function for the transformation, the construction of rasterized pyramidal shape (bigger at bottom and smaller at top) made of voxels can be regarded as an assignment problem, which is solved by Hungarian method. The study considered four moving strategies for pick and place, and compared their performance in terms of the transformational cost and computational time. As a result, LOS stands out as it achieves an appealing balance between these two metrics.

For pick and place operation on heterogeneous components, LEGO bricks are applied as an outstanding representation, since it has been investigated as LEGO construction problem. Starting from this problem, some of the heuristic algorithms in LEGO construction, i.e., to merge voxels into basic bricks are compared and analyzed. In this process, it's discovered that the transformation strategy can be directly applied in the process of merging. Therefore, two basic approaches have been proposed, namely rigid transform (RT), or layout first transform approach and flexible transform approach (FT), or transform first approach. Under each approach, three strategies were

investigated in difference of sequencing priority for moving bricks: from source to buffer, from buffer to target, from source to target and from stock to target. Reuse rate, cost and overall performance are studies to evaluate the performance of these strategies.

Moreover, the sequencing problem for chain structures is studied, which is operated as folding. Specifically, instead of folding 2D into 3D (origami), I look into the problem of folding 1D into 2D or 3D, which has more possibilities for automation in folding. Firstly, the problem of finding a 1D chain from 3D rasterized shape has been studied, which is solved as a Hamiltonian path finding problem either by prune and append method or division of voxels method. Then folding sequencing strategies have been proposed in a stream.

Strategies are designed to transform these problem into mathematical model, and mathematical principles such as Hungarian method for assignment problems and Hamiltonian path searching skills for traveling salesman problem are incorporated in these studies. Once the advanced building materials and tools are developed in construction and manufacturing industry, these sequencing and automation strategies can be directly put into applications.

6.2 Future Work

For future work, there are several factors which can further improve the sequencing work for automation in construction:

1.Force analysis within the structures such as finite element analysis (FEA) can be incorporated in the 3D model rasterization process, in order to confirm that the whole system is robust: not only the final state of the structure is stable, but the intermediate structures are stable as well;

2.Modularization is not considered in this study, which is more practical for material reuse. Similar to some applications in electrical or mechanical engineering, by

considering a group of voxels as a whole and defining some basic modulars with different functions, the reuse rate and cost can be significantly since the transformation is at a modular level;

Finally, I'd like to put an end to this thesis by copying a cartoon image from an article of the bibliography (Figure 6.1). With the development of artificial intelligence and advanced technical tools, I can foresee the construction, manufacturing as well as assembly work will be highly automatic in the future.

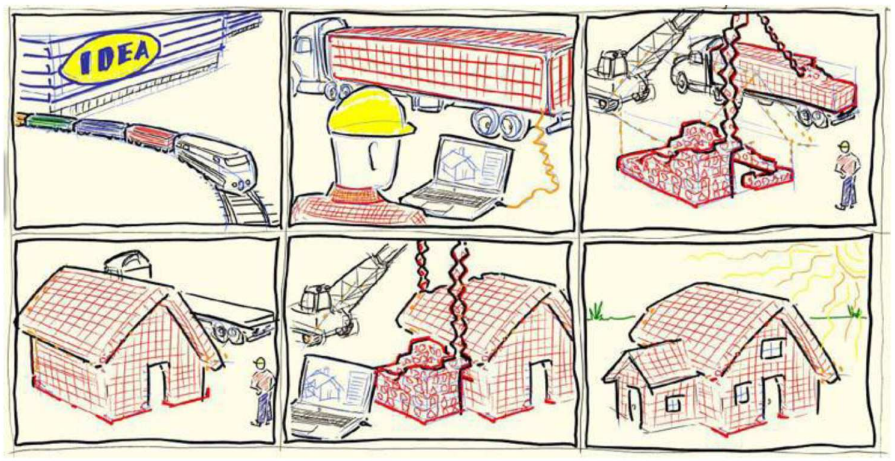

FIGURE 6.1: A cartoon demonstrating an idealized scenario for programmable matter as an architectural construction material, offering new construction and adaptive reuse of existing materials (Tibbits and Cheung, 2012).

Appendix A

Hamiltonian Path in a Graph[1]

A.1 Definition

Definition A.1.1 (Hamiltonian Path/Circuit) *Given a (connected) graph $G(V, E)$, find a path that starts at a vertex of a graph, passes through every vertex exactly once, and returns to another vertex. This path is called* ***Hamiltonian Path (HP)****. If the starting vertex returns to itself, a circuit rather than a path is found, which is called* ***Hamiltonian Circuit (HC)****.*

The problem of deciding whether a graph has a Hamiltonian circuit/path (and finding one) or not is NP-complete in the general case. We will later see though that for some specific types of graphs it is solvable in polynomial time. For weighted graphs a more important and practically useful challenge is to find a Hamiltonian path with the minimum sum of weights over the edges we pass. This path is called an optimal Hamiltonian path. The problem is well known as the Traveling Salesman Problem (TSP).

A.2 Existence

Before trying to find a Hamiltonian path or circuit it is useful to try to find whether the examined graph has one. This is an NP-Complete problem as well but for some specific types of graphs criteria that help us decide faster have been set. Here is a list of the most useful and commonly used:

[1]The content in this chapter is referred from articles (Itai, Papadimitriou, and Szwarcfiter, 1982; Dong Chen, Shen, and Topor, 2002), online sources ("Eulerian and Hamiltonian Paths Circuits"; *Eulerian and Hamiltonian Paths Circuits*) and book (Wilson and Watkins, 1990).

Definition A.2.1 *A directed graph G(V,E) such that for each pair $v_i, v_j \in V$ either (v_i, v_j) or (exclusive) $(v_i, v_j) \in E$ is called **Tournament** Graph. A Tournament is a digraph in which there is exactly one arc between any two vertices.*

Theorem A.2.1 *A tournament graph always contains a Hamilton path.*

The proof is a matter of induction.

For undirected graphs:

Theorem A.2.2 *Any complete or chain (all nodes have degree=2) graph has Hamiltonian circuits.*

Additionally, researchers have proved that any triangular planar graph can be tested in polynomial time and this resulted in a useful theorem:

Definition A.2.2 *A planar graph that will remain connected if we substract any 4 edges, but there is a set of 5 edges which being omitted results in a non-connected graph is called **4-connected**.*

Theorem A.2.3 *Any 4-connected planar graph contains a Hamilton circuit that can be found in polynomial time.*

Tutte proved that any 4-connected planar graph contains a Hamilton circuit. This circuit can be found it polynomial time. Also, Tutte proved that any graph obtained by deleting a vertex from a 4-connected planar graph contains a Hamilton circuit.

Unfortunately, in the general case we have to try to build a Hamilton path or circuit and decide that there does not exist any only after exhaustive search of the paths. But it can become even worse (Murphy's law) if we consider the fact that slight changes in the graph may result in a totally different result. For example, we can see (via exhaustive search) that the following graph (Figure A.1a) has a Hamiltonian cycle: the path AGFECDBA.

If we remove the edge BD we have the following graph (Figure A.1b), there is no Hamilton circuit as a result.

Fortunately, we can list a number of conditions that must be held in a graph that has a Hamilton circuit:

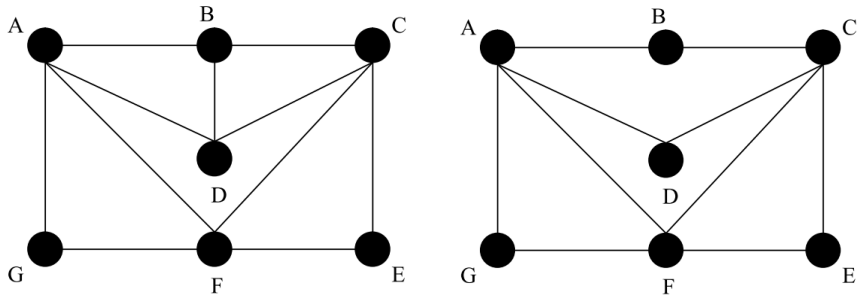

FIGURE A.1: (a) The graph has a Hamiltonian cycle: the path AGFECDBA. (b) Removing the edge BD results in no Hamilton circuit.

Necessary Conditions for Hamilton circuit Assume $G = (V, E)$ has more than two nodes.

1. No vertex of degree one. If a node has degree one, then the other endpoint of the edge incident to a must be visited at least twice in any circuit of G.

2. If a node has degree two, then both edges incident to it must be in any Hamilton circuit.

3. No smaller circuits contained in any Hamilton circuit (the start/endpoint of any smaller circuit would have to be visited twice).

4. There must exist a subgraph of G with the following properties:

    1. H contains every vertex of G;
    2. H is connected;
    3. H has the same number of edges as vertices;
    4. H has every node with degree 2.

Then, the subgraph H is the Hamilton circuit in the graph G. The problem is that the 4th condition can not be tested in polynomial time, but the 3 first conditions can be tested without much efforts. Furthermore, a very useful theorem and quite general (the only restriction is that each vertex should have $|E|/2$ adjacent vertices or more) was proved by Dirac:

Theorem A.2.4 (Dirac's Theorem) *If every vertex of a connected graph with 3 or more vertices is adjacent to at least half of the remaining vertices, then the graph has a Hamilton circuit.*

So we can see that even though we have a number of theorems for specific types of graphs the matter of existence of a Hamilton cycle is NP-complete in the general case.

A.3 Finding an Optimal Hamiltonian circuit/path.

As stated above, not only the problem is NP-complete, but we do not have any algorithm (even heuristic) to solve it in the general case as well. The only method is the exhaustive search which is "prohibited" when |V| and |E| become big. A more useful problem is the traveling salesman problem (TSP), which was originally set as follows:

Definition A.3.1 (TSP) *Given a list of cities and the distances between each pair of cities, what is the best possible route that visits each city and returns to the origin city?*

Here, best has the meaning of the minimum distance the salesman has to cover, but depending on the context given it may mean different things such as lower cost, minimum time etc.

We can represent each city by using a node, the way between any two cities as an edge between the respective nodes and the distance as the weight of the edge. The result is a complete graph, unless there is no way to go directly from one city to some other. In this case, the respective edge does not exist. The algorithm proposed is a brutal-force algorithm:
1. List all possible Hamilton circuits (of course exact reversals can be omitted);
2. Find the weight of each;
3. Choose (the) one with the smallest weight.
This algorithm always works, but its main backbone is its cost. The 1st step can only be solved by exhaustive search. For a computer doing 10,000 circuits/sec, it would take about 18 seconds to handle 10 vertices, 50 days to handle 15 vertices, 2 years for 16 vertices, 193,000 years for 20 vertices! This has made researchers change their approach to the problem and find approximation algorithms that may not produce the optimal result but get close to it and (more important) work in acceptable time limits.

Three such algorithms are given here: They all take as input a graph and return a near-optimal-Hamilton circuit (if one exists).

CHEAPEST LINK ALGORITHM (CLA)
1. Choose the edge with the smallest weight (the "cheapest" edge), randomly breaking

ties;

2. Keep choosing the "cheapest" edge unless it (a) closes a smaller circuit OR (b) results in 3 selected edges coming out of a single vertex.

Continue until the Hamilton Circuit is complete. An example is given in Figure A.2.

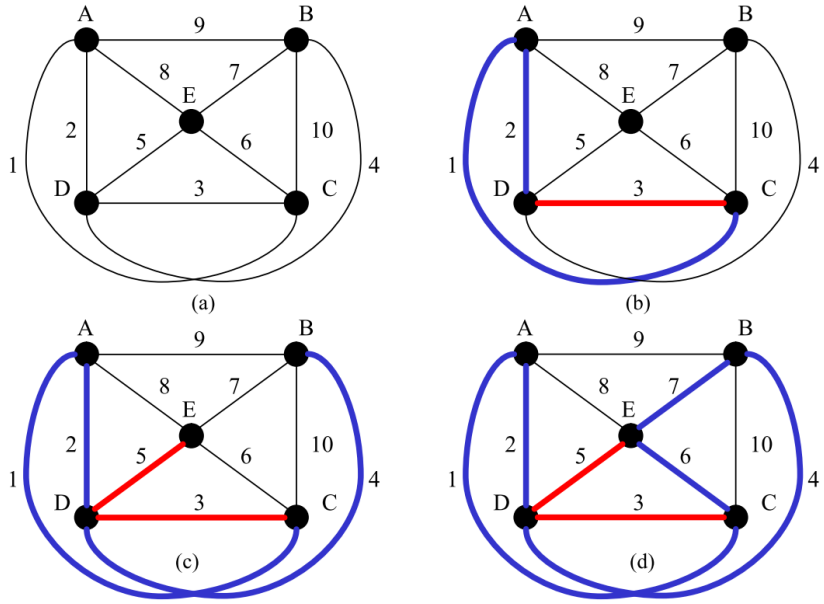

FIGURE A.2: Applying CLA algorithm: (a) We choose AC(min weight). Then, since no restriction will be validated we add AD. Now if we try to add DC we get a cycle ACDA (b) so we do not add it. Instead we take the next lower weighted edge DB and add it. If we add ED node D will have degree 3 (c), so we do not use it. Instead we add EC and EB successively so we get a Hamiltonian cycle ADBECA (d) with total cost 20.

NEAREST NEIGHBOR ALGORITHM (NNA)

1. Start at a vertex (think of it as your Home city);

2. Travel to the vertex that you haven't been to yet whose path has the smallest weight. If there is a tie, pick randomly;

3. Continue until you travel to all vertices;

4. Travel back to your starting vertex;

Figure A.3 exemplifies how this algorithm is applied for a graph.

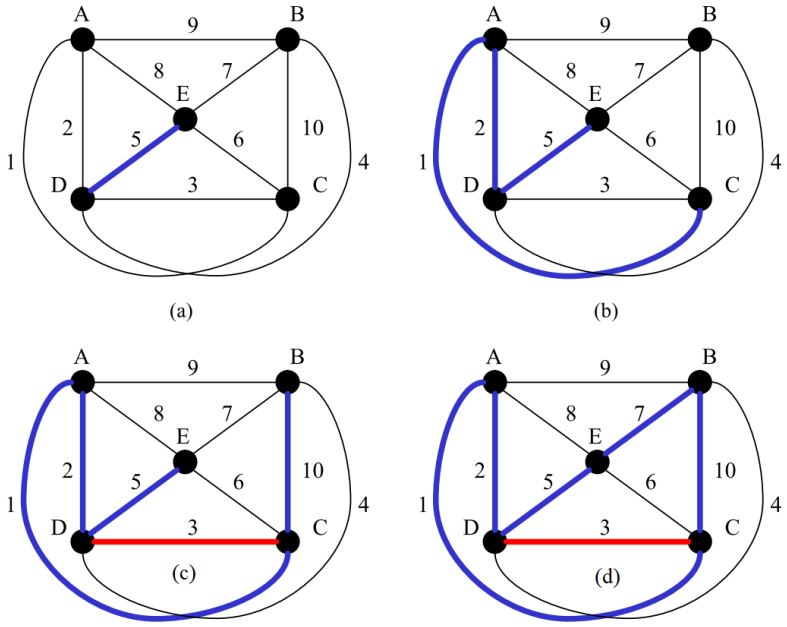

FIGURE A.3: Applying NNA algorithm: Randomly we choose E as the initial vertex. We choose the edge with the lower cost starting from E (a) which is ED. Same way we add DA and AC and reach (b). Here we can not use the edge with the lower cost (CD) since D has already been visited so we travel to B through CB(c) and finally reach the circuit EDACBE (d) with total cost 25 (worse than the CLA).

REPETITIVE NEAREST NEIGHBOR ALGORITHM (RNNA)

1. DO NNA for each vertex (city);
2. Choose the best solution (smallest weight);
3. If necessary, rewrite this solution with a particular starting vertex (Home);

It uses the previous algorithm, which depends on the choice of the initial node. That's why we apply it for every initial node possible and choose the best circuit among the ones produced.

A similar problem is to find a Hamiltonian path given the initial and final nodes. This would help the salesman end his journey at his home and not the city he was at the beginning.

There exists a constructive algorithm (Algorithm 1) that gives us the optimal Hamilton path between two nodes a (initial) and p (terminal). This algorithm tries to find the

optimal path between any two vertices visiting any number ($\leq |V|-2$) of vertices starting from 0 (no intermediates) and adding one node to the (new) path at the time. For instance, in order to find the minimum path between a and p that has 3 intermediate nodes we find all the paths from any node k to p with 2 intermediate nodes ($\neq a, p$) and add (a, k) and choose the minimum.

Algorithm 1 Optimal Hamilton Path (w, a, p)

Input: w: matrix of weights, a: initial node, p: final node and N= | V |
Output: The Optimal Hamilton Path;
S: subset $\{1, 2, \ldots, N\}$; Array $H[N][N][S]$; int i, j, s, k

1: **for** $i, j \leftarrow 1, N$ **do**
2: $H[i, j, \emptyset] \leftarrow w(i, j)$
3: **end for**
4: **for** $s \leftarrow 1, N-1$ **do**
5: **for** $i, j \leftarrow 1, N$ **do**
6: **for** each $S \subseteq [1 \ldots N] - \{i, j\}$ with size s **do**
7: $H[i, j, S]$=min$[d(i, k) + H(k, j, S - \{k\})]$
8: **end for**
9: **end for**
10: **end for**
11: return $H[a, p, \{1 \ldots N\} - \{a, p\}]$

The algorithm runs in $O(2^N)$ time significantly lower than $O(N!)$ which gives us the ability to work with larger graphs in appropriate time. The reason we have this complexity has to do with the number of the paths we have to calculate in step 6 before we select the minimum among them.

A.4 Hamilton Path in Grid (Lattice) Graphs

Definition A.4.1 *An infinite graph G^∞ is a graph whose vertex set $V(G^\infty)$ consists of all vertices on the plane with integer coordinates and in which two vertices are connected by an edge iff the Euclidean distance between them is equal to 1.*

Definition A.4.2 *The coordinates of a vertex v are denoted by v_x, v_y and v_z. Vertex v is called even if $(v_x + v_y + v_z)$ mod 2 = 0, or odd otherwise.*

Definition A.4.3 *A grid graph G is a finite, node-induced subgraph of G^∞ whose vertex set is $V(G) = \{v | 1 \leq v_x \leq m, 1 \leq v_y \leq n, and 1 \leq v_z \leq l, \}$, where m, n and l are integers.*

Definition A.4.4 *Two different vertices v and $v\prime$ in G are called color-compatible if*
(1) The number of even vertices equals the number of odd vertices ($|V_{odd}| = |V_{even}|$, G is even), and s and t have different color; or
(2) G is not even, e.g. the number of odd vertex is greater than the number of even vertex ($|V_{odd}| = |V_{even}| + 1$), and s and t are colored by the majority color ($s, t \in V_{odd}$).

Since two adjacent vertices must have different colors, the vertices of any path (or cycle) alternate between two colors. Thus, we have the following two simple lemmas.

Lemma A.4.1 *G has a HC iff. it is even-sized and $m, n, x > 1$.*

Lemma A.4.2 *If G has a HP between two given vertices s and t, then s and t must be color-compatible.*

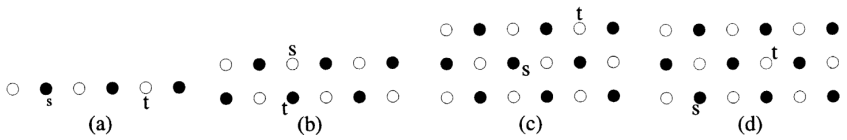

FIGURE A.4: Graphs in which there is no Hamiltonian path from vertex s to t.

We denote a Hamiltonian path between s and t in G by $HP(G, s, t)$; We say $HP(M, s, t)$ existent if there is a Hamiltonian path between s and t in G. From Lemma A.4.2, we conclude that the color-compatibility of s and t is a necessary condition for the existence of a Hamiltonian path on G. However, there are three cases in which $HP(M, s, t)$ are non-existent even if s and t are color-compatible:
C1: G is a one-mesh, and either s or t is not a corner vertex (Figure A.4(a)).
C2: G is a two-mesh, and st is an edge which is not a boundary edge (Figure A.4(b)).
C3: n = 3, t=1 and m is even, s is black and t is white, and
1. $s_y = 2$ and $t_x > s_x$ (Figure A.4(c)); or
2. $s_y \neq 2$ and $t_x \leq s_x + 1$ (Figure A.4(d)).

The following theorem has been proved (Itai, Papadimitriou, and Szwarcfiter, 1982).

Theorem A.4.3 *$HP(M, s, t)$ is existent if and only if s and t are color-compatible and it is not in any of the cases C1 to C3.*

Theorem A.4.4 *The existence of any $HP(M, s, t)$ can be determined in constant time.*

Algorithm for HP in Grid Graph A sequential and a parallel algorithm have been described in the work of Dong Chen et al. (Dong Chen, Shen, and Topor, 2002).

Appendix B

Assignment Problem and Hungarian Method: An Implementation[1]

B.1 Mathematical Formulation of Assignment Problem

Consider the problem of assignment of a company which has n workers each using different time to finish n different jobs and one worker can only be assigned to do only one job. The objective is to minimize the total time of assignment. The cost matrix for this problem is given in Table B.1. This cost matrix is same as that of a transportation problem except that the availability at each of the machines is unity and the requirement at each of the destinations is also unity. In Table B.1, t_{ij} is the cost associated with assigning the ith machine to the jth job. To formulate the assignment problem in a mathematical way, we define the activity variables

$$x_{ij} = \begin{matrix} 1, \text{if worker i is assigned to job j} \\ 0, \text{otherwise} \end{matrix} \tag{B.1}$$

Then the mathematical model for the assignment problem can be stated as

[1]The content in this chapter is referred from online sources (*Transportation and Assignment Problems*; *The Assignment Problem and the Hungarian Method*; *The Hungarian algorithm: An example*; *The Hungarian algorithm: An example*) and book (Burkard, Dell'Amico, and Martello, 2009).

$$\text{Minimize } Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{B.2}$$

$$\text{subject to } \sum_{j=1}^{n} x_{ij} = 1 \text{ for all } i \tag{B.3}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for all } j \tag{B.4}$$

$$\text{and } x_{ij} = 0 \text{ or } 1 \text{ for all } i \text{ and } j \tag{B.5}$$

TABLE B.1: The cost for n workers to finish n jobs. Cost is defined as time here.

Workers	Jobs
	J_1 J_2 ... J_n
W_1	c_{11} c_{12} ... c_{1n}
W_2	c_{21} c_{22} ... c_{2n}
⋮	⋮⋮⋮⋮
W_n	c_{n1} c_{n2} ... c_{nn}

B.2 Hungarian Method for Solving Assignment Problem

The Hungarian method is an efficient method for finding the optimal solution of an assignment problem (Kuhn, 1955; Frank, 2005) with a given cost matrix. The method works on the principle of reducing the given cost matrix to a matrix of opportunity costs (Theorem B.2.1). Opportunity costs show the relative penalties associated with assigning a machine to a job. Hungarian method reduces the cost matrix to the extent of having at least one zero in each row and column so as to make optimal assignment. The following 6-step algorithm is a modified form of the original Hungarian Algorithm (Munkres' implementation). This algorithm describes to the manual manipulation of a two-dimensional matrix by starring and priming zeros and by covering and uncovering rows and columns. This is because, at the time of publication (in 1957), few people had access to a computer and the algorithm was exercised by hand.

Theorem B.2.1 (Reduction Theorem) *In an assignment problem, adding or subtracting a constant to every element of a row or column of the cost matrix has no influence on the optimal assignment plan. In other words, if* $x_{ij} = x_{ij}^{\star}$ *minimizes* $Z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$ *defined in Equation B.2 with requirements on Equation B.3 to B.5,* $x_{ij}^{\star}$ *also minimizes* $Z' = \sum_{i=1}^{n}\sum_{j=1}^{n} c'_{ij}x_{ij}$ *where* $c\prime_{ij} = c_{ij} - u_i - v_j$ *for all* $i, j = 1, 2, \ldots, n$ *and* $u_i, v_j \in \Re$.

Step 0 Create an $n \times m$ matrix called the cost matrix in which each element represents the cost of assigning one of n workers to one of m jobs. Rotate the matrix so that there are at least as many columns as rows and let k=min(n, m).

Step 1 For each row of the matrix, find the smallest element and subtract it from every element in its row. Go to **Step 2**.

Step 2 Find a zero (Z) in the resulting matrix. If there is no starred zero in its row or column, star Z. Repeat for each element in the matrix. Go to **Step 3**.

Step 3 Cover each column containing a starred zero. If K columns are covered, the starred zeros describe a complete set of unique assignments. In this case, Go to **DONE**, otherwise, Go to **Step 4**.

Step 4 Find a noncovered zero and prime it. If there is no starred zero in the row containing this primed zero, Go to **Step 5**. Otherwise, cover this row and uncover the column containing the starred zero. Continue in this manner until there are no uncovered zeros left. Save the smallest uncovered value and Go to **Step 6**.

Step 5 Construct a series of alternating primed and starred zeros as follows. Let Z_0 represent the uncovered primed zero found in **Step 4**. Let Z_1 denote the starred zero in the column of Z_0 (if any). Let Z_2 denote the primed zero in the row of Z_1 (there will always be one). Continue until the series terminates at a primed zero that has no starred zero in its column. Unstar each starred zero of the series, star each primed zero of the series, erase all primes and uncover every line in the matrix. Return to **Step 3**.

Step 6 Add the value found in **Step 4** to every element of each covered row, and subtract it from every element of each uncovered column. Return to **Step 4** without altering any stars, primes, or covered lines.

DONE Assignment pairs are indicated by the positions of the starred zeros in the cost matrix. If $C(i, j)$ is a starred zero, then the element associated with row i is assigned to the element associated with column j.

Some of these descriptions require careful interpretation. In Step 4, for example, the possible situations are, that there is a noncovered zero which get primed and if there is no starred zero in its row the program goes onto Step 5. The other possible way out of Step 4 is that there are no noncovered zeros at all, in which case the program goes to Step 6.

An implementation of C# code as well as an example can be found here: *The Hungarian algorithm: An example*, as shown in Figure B.1.

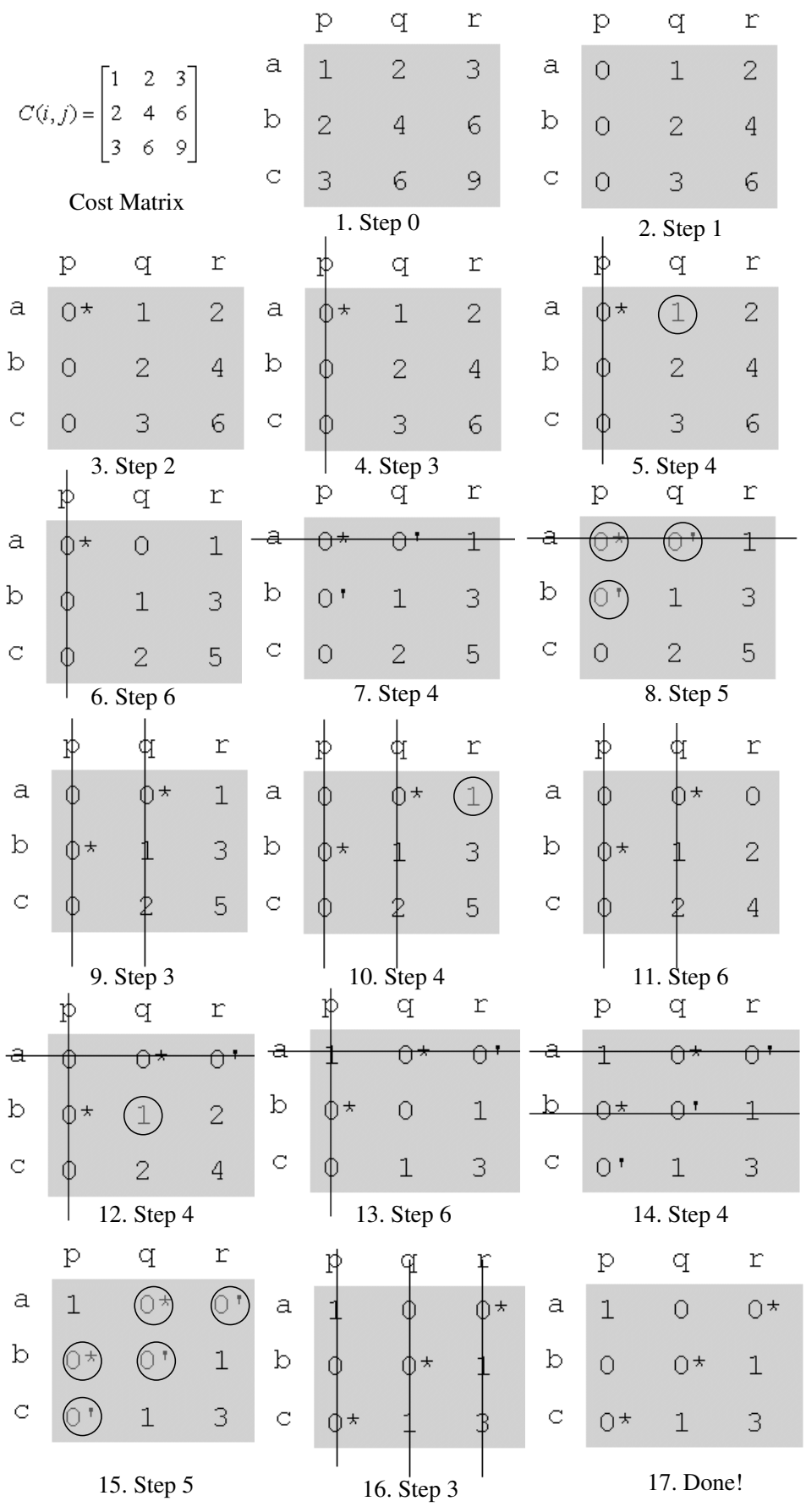

FIGURE B.1: An Example Execution of Hungarian Method (*The Hungarian algorithm: An example*).

Bibliography

Accidents at work statistics (2014). http://ec.europa.eu/eurostat/statistics-explained/index.php/Accidents_at_work_statistics.

Addis, Bill (2012). *Building with reclaimed components and materials: a design handbook for reuse and recycling*. Routledge.

Ahmadzadeh, Hossein and Ellips Masehian (2015). "Modular robotic systems : Methods and algorithms for abstraction , planning , control , and synchronization". In: *Artificial Intelligence* 223, pp. 27–64. ISSN: 0004-3702. DOI: 10.1016/j.artint.2015.02.004. URL: http://dx.doi.org/10.1016/j.artint.2015.02.004.

Ahn, Hee-Kap, Siu-Wing Cheng, and Iris Reinbacher (2013). "Maximum overlap of convex polytopes under translation". In: *Computational Geometry* 46, pp. 552–565. DOI: 10.1016/j.comgeo.2011.11.003.

Ahn, Hee-Kap et al. (2014). "Overlap of convex polytopes under rigid motion". In: *Computational Geometry* 47, pp. 15–24. DOI: 10.1016/j.comgeo.2013.08.001.

Anderson, David et al. (2000). "Tangible interaction+ graphical interpretation: a new approach to 3D modeling". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., pp. 393–402. DOI: 10.1145/344779.344960.

Ardiny, Hadi, Stefan Witwicki, and Francesco Mondada (2015). "Construction automation with autonomous mobile robots: A review". In: *Robotics and Mechatronics (ICROM), 2015 3rd RSI International Conference on*. IEEE, pp. 418–424. DOI: 10.1109/ICRoM.2015.7367821.

Augugliaro, Federico et al. (2014). "The Flight Assembled Architecture Installation: Cooperative Contruction with Flying Machines". In: *Control Systems, IEEE* 34.4, pp. 46–64. DOI: 10.1109/MCS.2014.2320359.

Ayob, Masri, Peter Cowling, and Graham Kendall (2002). "Optimisation for surface mount placement machines". In: *Industrial Technology, 2002. IEEE ICIT'02. 2002 IEEE International Conference on*. Vol. 1. IEEE, pp. 498–503.

Ayob, Masri and Graham Kendall (2008). "A survey of surface mount device placement machine optimisation: Machine classification". In: *European Journal of Operational Research* 186.3, pp. 893–914. DOI: 10.1016/j.ejor.2007.03.042.

Bachrach, J, V Zykov, and S Griffith (2009). "Folding Arbitrary 3D Shapes with Space Filling Chain Robots : Reverse Explosion Approach to Folding Sequence Design". In: *Power* 1.

Balaguer, Carlos (2000). "Open issues and future possibilities in the EU construction automation". In: *Proceedings of the IAARC International Symposium on Robotics and Automation, Taipei, Taiwan*. Citeseer.

Balkcom, Devin J. and Matthew T. Mason (2008). "Robotic origami folding". In: *The International Journal of Robotics Research* 27.5, pp. 613–627. ISSN: 0278-3649. DOI: 10.1177/0278364908090235. URL: http://journals.sagepub.com/doi/10.1177/0278364908090235.

Bentzen, B (2000). "SMD placement". In: *the SMT in FOCUS*.

Bidgoli, Ardavan (2015). "Towards an integrated design making approach in architectural robotics". In:

Bock, T et al. (1996). "Automatic generation of the controlling-system for a wall construction robot". In: *Automation in construction* 5.1, pp. 15–21. DOI: 10.1016/0926-5805(95)00014-3.

Bock, Thomas and Silke Langenberg (2014). "Changing Building Sites: Industrialisation and Automation of the Building Process". In: *Architectural Design* 84.3, pp. 88–99.

Burkard, Rainer E, Mauro Dell'Amico, and Silvano Martello (2009). *Assignment Problems, Revised Reprint*. Siam. URL: http://bookstore.siam.org/otr106/.

Canstruction. https://www.canstruction.org/. a unique charity art exhibition and event.

Castano, Andres, R Chokkalingham, and Peter M Will (2000). "Autonomous and Self-Sufficient CONRO Modules for Reconfigurable Robots." In: *DARS*, pp. 155–164.

Castro-Lacouture, Daniel (2009). "Construction automation". In: *Springer Handbook of Automation*. Springer, pp. 1063–1078.

Celli, Paolo and Stefano Gonella (2015). "Manipulating waves with LEGO bricks: A versatile experimental platform for metamaterial architectures". In: *Applied Physics Letters* 107.8, p. 081901. DOI: 10.1063/1.4929566.

Chang, Zongyu et al. (2002). "A new method to mechanism kinematic chain isomorphism identification". In: *Mechanism and Machine Theory* 37.4, pp. 411–417. ISSN: 0094114X. DOI: 10.1016/S0094-114X(01)00084-2.

Cheung, Kenneth C. et al. (2011). "Programmable assembly with universally foldable strings (moteins)". In: *IEEE Transactions on Robotics* 27.4, pp. 718–729. ISSN: 15523098. DOI: 10.1109/TRO.2011.2132951.

Chiang, Chih-Jung and Gregory S Chirikjian (2001). "Modular robot motion planning using similarity metrics". In: *Autonomous Robots* 10.1, pp. 91–106. DOI: 10.1023/A:1026552720914.

Cignoni, Paolo et al. (2008). "MeshLab: an Open-Source Mesh Processing Tool". In: *Eurographics Italian Chapter Conference*. Ed. by Vittorio Scarano, Rosario De Chiara, and Ugo Erra. The Eurographics Association. ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

Cohen-Or, Daniel and Arie Kaufman (1997). "3D line voxelization and connectivity control". In: *IEEE Computer Graphics and Applications* 17.6, pp. 80–87.

D'andrea, Raffaello et al. (2010). *Method and system for transporting inventory items*. US Patent 7,826,919.

De Berg, Mark et al. (1998). "Computing the maximum overlap of two convex polygons under translations". In: *Theory of computing systems* 31.5, pp. 613–628. DOI: 10.1007/BFb0009488.

DE SOUZA, ROBERT and WU LIJUN (1994). "CPS: A productivity tool for component placement in multi-head concurrent operation PCBA machines". In: *Journal of Electronics Manufacturing* 4.02, pp. 71–79.

De Souza, Robert and Wu Lijun (1995). "Intelligent optimization of component onsertion in multi-head concurrent operation PCBA machines". In: *Journal of Intelligent Manufacturing* 6.4, pp. 235–243.

Deshpande, Vikram S, Norman A Fleck, and Michael F Ashby (2001). "Effective properties of the octet-truss lattice material". In: *Journal of the Mechanics and Physics of Solids* 49.8, pp. 1747–1769.

Dierichs, Karola and Achim Menges (2016). "Towards an aggregate architecture: designed granular systems as programmable matter in architecture". In: *Granular Matter* 18.2, p. 25.

— (2017). "Granular Construction: Designed Particles for Macro-Scale Architectural Structures". In: *Architectural Design* 87.4, pp. 88–93.

Diez, Ramiro et al. (2000). "Autmod3: an automatic 3D modularization system". In: *17th ISARC, Taipei,(Taiwan)*, pp. 1033–1038.

Ding, S, MA Mannan, and Aun Neow Poo (2004). "Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces". In: *Computer-Aided Design* 36.13, pp. 1281–1294.

Ding, Xilun and Shengnan Lu (2013). "Fundamental reconfiguration theory of chain-type modular reconfigurable mechanisms". In: *Mechanism and Machine Theory*. ISSN: 0094114X. DOI: 10.1016/j.mechmachtheory.2013.08.011.

Dong Chen, Shao, Hong Shen, and Rodney Topor (2002). "An efficient algorithm for constructing Hamiltonian paths in meshes". In: *Parallel Computing* 28.9, pp. 1293–1305. ISSN: 01678191. DOI: 10.1016/S0167-8191(02)00135-7.

Duncan, Sean C (2011). "Minecraft, beyond construction and survival". In: *Well Played: a journal on video games, value and meaning* 1.1, pp. 1–22.

Esmaeilian, Behzad, Sara Behdad, and Ben Wang (2016). "The evolution and future of manufacturing: A review". In: *Journal of Manufacturing Systems* 39, pp. 79–100.

Feczko, Jacek et al. (2015). "Review of the modular self reconfigurable robotic systems". In: *Robot Motion and Control (RoMoCo), 2015 10th International Workshop on*. IEEE, pp. 182–187. DOI: 10.1109/RoMoCo.2015.7219733.

Fei, L. J. and D. Sujan (2013). "Origami Theory and its Applications : A Literature Review". In: *World Academy of Science, Engineering and Technolofy: International Journal od Social, Management, Economics and Business Engineering* 7.1, pp. 113–117.

Feijs, Loe and Roel De Jong (1998). "3D visualization of software architectures". In: *Communications of the ACM* 41.12, pp. 73–78. DOI: 10.1145/290133.290151.

Fink, Jonathan et al. (2011). "Planning and control for cooperative manipulation and transportation with aerial robots". In: *The International Journal of Robotics Research* 30.3, pp. 324–334. DOI: 10.1007/978-3-642-19457-3.

Fiore, Albie (1981). *Shaping Rubik's Snake*. Penguin Australia.

Follett, Ken (2010). *The pillars of the earth*. Penguin.

Frank, András (2005). "On Kuhn's Hungarian Method−A tribute from Hungary". In: *Naval Research Logistics (NRL)* 52.1, pp. 2–5. ISSN: 1520-6750. DOI: 10.1002/nav.20056.

Frohm, Jörgen et al. (2008). "Levels of automation in manufacturing". In: *Ergonomia*.

Fukuda, Komei and Takeaki Uno (2006). "Algorithms for maximizing the volume of intersection of polytopes". In: *Proc. 22nd European Workshop on Computational Geometry (EWCG 2006)*. Citeseer, pp. 197–200.

— (2007). "Polynomial time algorithms for maximizing the intersection volume of polytopes". In: *Pacific Journal of Optimization* 3.1, pp. 37–52. DOI: 10.1.1.610.6389.

Fulford, Richard and Craig Standing (2014). "Construction industry productivity and the potential for collaborative practice". In: *International Journal of Project Management* 32.2, pp. 315–326.

Gao, Wei et al. (2013). "Kinetogami: A Reconfigurable, Combinatorial, and Printable Sheet Folding". In: *Journal of Mechanical Design* 135.11, p. 111009. ISSN: 1050-0472. DOI: 10.1115/1.4025506. URL: http://mechanicaldesign.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4025506.

Gassel, Frans van (1996). "Mechanization and Automation by the Manufacturing of Removable Modular Buildings". In: *13th ISARC, Tokyo (Japan)*, pp. 1019–1026.

Gilpin, Kyle, Ara Knaian, and Daniela Rus (2010). "Robot pebbles: One centimeter modules for programmable matter through self-disassembly". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, pp. 2485–2492.

Gilpin, Kyle, Kent Koyanagi, and Daniela Rus (2011). "Making self-disassembling objects with multiple components in the robot pebbles system". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, pp. 3614–3621.

Gilpin, Kyle et al. (2008). "The International Journal of Miche : Modular Shape". In: *The International Journal of Robotics Research*. DOI: 10.1177/0278364907085557.

Giri, Bibhas C. *Transportation and Assignment Problems*. URL: http://epgp.inflibnet.ac.in/epgpdata/uploads/epgp_content/mathematics/14._operations_research/08._transportation_and_assignment_problems__assignment_problem_and_its_solution_by_hungarian_method,_and_travelling_salesman_problem/et/9225_et_et.pdf.

Glynn, Ruairi and Bob Sheil (2011). *Fabricate: Making Digital Architecture*. Riverside Architectural Press.

Gothelf, Kurt V (2012). "LEGO-like DNA structures". In: *Science* 338.6111, pp. 1159–1160. DOI: 10.1126/science.1229960.

Gower, R, A Heydtmann, and H Petersen (1998). "LEGO: automated model construction". In: *32nd European Study Group with Industry Final Report*, pp. 81–94.

Griffith, Saul Thomas and Joseph Jacobson (2004). "Growing Machines". In: *Media* Media Arts. URL: http://dspace.mit.edu/bitstream/handle/1721.1/28780/60129661.pdf?sequence=1.

Grotzinger, Stephen (1992). "Feeder assignment models for concurrent placement machines". In: *IIE transactions* 24.4, pp. 31–46.

G.Tollis, Ioannis. *Eulerian and Hamiltonian Paths Circuits*. URL: http://www.csd.uoc.gr/~hy583/reviewed_notes/euler.pdf.

Gupta, Ankit et al. (2012). "DuploTrack: a real-time system for authoring and guiding duplo block assembly". In: *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pp. 389–402. DOI: 10.1145/2380116.2380167.

Hagis, Christopher (2003). *History of Robots*.

Helm, Volker et al. (2012). "Mobile robotic fabrication on construction sites: DimRob". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 4335–4341. DOI: 10.1109/IROS.2012.6385617.

Hiller, Jonathan and Hod Lipson (2009). "Design and analysis of digital materials for physical 3D voxel printing". In: *Rapid Prototyping Journal* 15.2, pp. 137–149. DOI: 10.1108/13552540910943441.

Ho, William and Ping Ji (2005). "A genetic algorithm to optimise the component placement process in PCB assembly". In: *The International Journal of Advanced Manufacturing Technology* 26.11-12, pp. 1397–1401. DOI: 10.1007/s00170-004-2132-5.

Hong, Jhen-Yao et al. (2016). "Inner engraving for the creation of a balanced lego sculpture". In: *The Visual Computer* 32.5, pp. 569–578. DOI: 10.1007/s00371-015-1072-4.

Howerter, Ryan. *LEGO color chart*. URL: http://swooshable.com/parts/colors.

Hu, Ruizhen et al. (2014). "Approximate pyramidal shape decomposition." In: *ACM Trans. Graph.* 33.6, pp. 213–1.

Huang, Jian et al. (1998). "An accurate method for voxelizing polygon meshes". In: *Volume Visualization, 1998. IEEE Symposium on*. IEEE, pp. 119–126. DOI: 10.1145/288126.288181.

Ichbiah, Daniel (2005). *Robots: From science fiction to technological revolution*. Harry N. Abrams.

Iguchi, Kazumoto (1998). "a Toy Model for Understanding the Conceptual". In: 12.13, pp. 499–506.

Itai, Alon, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter (1982). "Hamilton Paths in Grid Graphs". In: *SIAM Journal on Computing* 11.4, pp. 676–686. ISSN: 0097-5397. DOI: 10.1137/0211056. URL: http://epubs.siam.org/doi/10.1137/0211056.

Ix, Frank Dachille and Arie Kaufman (2000). "Incremental triangle voxelization". In: *Proceedings of Graphics Interface*, pp. 205–212.

Jan, Matej 'Retro' (2016). *Pixels and voxels, the long answer*. https://medium.com/retronator-magazine/pixels-and-voxels-the-long-answer-5889ecc18190.

Jiménez, Juan J and Rafael J Segura (2008). "Collision detection between complex polyhedra". In: *Computers & Graphics* 32.4, pp. 402–411.

Kalpakjian, Serope and Steven R Schmid (2014). *Manufacturing engineering and technology*. Pearson Upper Saddle River, NJ, USA.

Kanade, Takeo (1980). "A theory of Origami world". In: *Artificial Intelligence* 13.3, pp. 279–311. ISSN: 00043702. DOI: 10.1016/0004-3702(80)90004-1.

Kangari, Roozbeh and Tetsuji Yoshida (1990). "Automation in construction". In: *Robotics and Autonomous Systems* 6.4, pp. 327–335.

Kaufman, Arie, Daniel Cohen, and Roni Yagel (1993). "Volume graphics". In: *Computer* 26.7, pp. 51–64.

Ke, Yonggang et al. (2012). "Three-dimensional structures self-assembled from DNA bricks". In: *Science* 338.6111, pp. 1177–1183. DOI: 10.1126/science.1227268.

Khoo, LP and KM Loh (2000). "A genetic algorithms enhanced planning system for surface mount PCB assembly". In: *The International Journal of Advanced Manufacturing Technology* 16.4, pp. 289–296.

Kim, Jae Woo, Kyung Kyu Kang, and Ji Hyoung Lee (2014). "Survey on Automated LEGO Assembly Construction". In: *WSCG 2014 Conference on Computer Graphics, Visualization and Computer Vision*. Václav Skala-UNION Agency.

Kohler, Matthias, Fabio Gramazio, and J Willmann (2014). *The robotic touch: how robots change architecture*.

Kuhn, Harold W (1955). "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2, pp. 83–97. DOI: 10.1002/nav.3800020109.

Leonhard. "Eulerian and Hamiltonian Paths Circuits". In: 1 (), pp. 1–19. URL: http://www.csd.uoc.gr/{~}hy583/reviewed{_}notes/euler.pdf.

Leyh, Werner (1995). "Experiences with the construction of a building assembly robot". In: *Automation in construction* 4.1, pp. 45–60. DOI: 10.1016/0926-5805(94)00034-K.

Lindsey, Quentin, Daniel Mellinger, and Vijay Kumar (2011). "Construction of cubic structures with quadrotor teams". In: *Proc. Robotics: Science & Systems VII*, pp. 177–184. DOI: 10.15607/RSS.2011.VII.025.

Linner, Thomas (2013). "Automated and Robotic Construction: Integrated Automated Construction Sites". PhD thesis. München Technische Universität München.

Linner, Thomas and Thomas Bock (2012). "Evolution of large-scale industrialisation and service innovation in Japanese prefabrication industry". In: *Construction Innovation* 12.2, pp. 156–178.

Luo, Sheng-Jie et al. (2015). "Legolization: optimizing lego designs". In: *ACM Transactions on Graphics (TOG)* 34.6, p. 222. DOI: 10.1145/2816795.2818091.

Lupashin, Sergei et al. (2014). "A platform for aerial robotics research and demonstration: The Flying Machine Arena". In: *Mechatronics* 24.1, pp. 41–54. DOI: 10.1016/j.mechatronics.2013.11.006.

Magnenat, Sté phane, Roland Philippsen, and Francesco Mondada (2012). "Autonomous construction using scarce resources in unknown environments". In: 33.4, pp. 467 – 485.

Mirjan, A et al. (2013). "Architectural fabrication of tensile structures with flying machines". In: *Green Design, Materials and Manufacturing Process, CRC Press, Boca Raton FL*, pp. 513–518. DOI: 10.1201/b15002-99.

Mitani, Jun (2016). *3D origami art*, p. xi. ISBN: 978-1-4987-6535-0.

Miyatake, Yasuyoshi and Roozbeh Kangari (1993). "Experiencing computer integrated construction". In: *Journal of Construction Engineering and Management* 119.2, pp. 307–322.

Mohan, Yogeswaran and SG Ponnambalam (2009). "An extensive review of research in swarm robotics". In: *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*. IEEE, pp. 140–145.

Moselhi, O (1998). "Robots and Automated Machines in Construction". In: *The International Association for Automation and Robotics in Construction, Watford, England, UK*.

Moses, Matthew S et al. (2014). "An architecture for universal construction via modular robotic components". In: *Robotics and Autonomous Systems* 62.7, pp. 945–965. DOI: 10.1016/j.robot.2013.08.005.

Mueller, Stefanie et al. (2014). "faBrickation: fast 3D printing of functional objects by integrating construction kit building blocks". In: *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, pp. 3827–3834. DOI: 10.1145/2559206.2574779.

Murata, Satoshi and Haruhisa Kurokawa (2007). "Self-reconfigurable robots". In: *Robotics & Automation Magazine, IEEE* 14.1, pp. 71–78. DOI: 10.1109/MRA.2007.339607.

Murata, Satoshi et al. (2002). "M-TRAN: Self-reconfigurable modular robotic system". In: *IEEE/ASME transactions on mechatronics* 7.4, pp. 431–441.

Navarro, Iñaki and Fernando Matía (2012). "An introduction to swarm robotics". In: *ISRN Robotics* 2013.

Ono, Smiaki, Andre Alexis, Youngha Chang, et al. (2013). "Automatic generation of LEGO from the polygonal data". In: *IWAIT2013 in Nagoya*, pp. 262–267.

Pamecha, Amit, Imme Ebert-Uphoff, and Gregory S Chirikjian (1997). "Useful metrics for modular robot motion planning". In: *Robotics and Automation, IEEE Transactions on* 13.4, pp. 531–545. DOI: 10.1109/ROBOT.1996.503816.

Pan, Wei, Lujie Chen, and Stylianos Dritsas (2017). "Pick-and-place process sequencing for transformation of rasterized 3D structures". In: *Automation in Construction* 75, pp. 56–64.

Pasek, Zbigniew J and Derek Yip-Hoi (2005). "Lego Factory: An Educational CIM Environment for Assembly". In: *Proceedings of the 2005 American Society for Engineering Education Annual Conference & Exposition*. Vol. 10, p. 1.

Petrovic, Pavel (2001). "Solving LEGO brick layout problem using Evolutionary Algorithms". In: *Proceedings to Norwegian Conference on Computer Science*. DOI: 10.1.1.16.4146.

Prasad, Ray (2013). *Surface mount technology: principles and practice*. Springer Science & Business Media.

Pritschow, G et al. (1996). "Technological aspects in the development of a mobile bricklaying robot". In: *Automation in Construction* 5.1, pp. 3–13. DOI: 10.1016/0926-5805(95)00015-1.

Rihani, Rami A and Leonhard E Bernold (1994). "Computer integration for robotic masonry". In: *Computer-Aided Civil and Infrastructure Engineering* 9.1, pp. 61–67.

Romanishin, John W, Kyle Gilpin, and Daniela Rus (2013). "M-blocks: Momentum-driven, magnetic modular robots". In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 4288–4295.

Roper, William E (2006). "Strategies for building material reuse and recycle". In: *International journal of environmental technology and management* 6.3-4, pp. 313–345.

Roudaut, Anne et al. (2016). "Cubimorph: designing modular interactive devices". In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 3339–3345.

Safiuddin, Md et al. (2010). "Utilization of solid wastes in construction materials". In: *International Journal of Physical Sciences* 5.13, pp. 1952–1963.

Saidi, Kamel S, Thomas Bock, and Christos Georgoulas (2016). "Robotics in construction". In: *Springer handbook of robotics*. Springer, pp. 1493–1520.

Salemi, Behnam, Mark Moll, and Wei-Min Shen (2006). "SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, pp. 3636–3641.

Schoessler, Philipp et al. (2015). "Kinetic Blocks: Actuated Constructive Assembly for Interaction and Display". In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST'15. ACM, pp. 341–349. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807453.

Smal, Eugene (2008). "Automated brick sculpture construction". PhD thesis. Stellenbosch: Stellenbosch University.

Sramek, Milos (1996). "Visualization of volumetric data by ray tracing". In:

Strauss, Rudolf (1994). *Surface mount technology*. Digital Press.

Tam, Vivian WY and Chi Ming Tam (2006). "A review on the viable technology for construction waste recycling". In: *Resources, conservation and recycling* 47.3, pp. 209–221.

Testuz, Romain, Yuliy Schwartzburg, and Mark Pauly (2013). "Automatic Generation of Constructable Brick Sculptures". In: *Eurographics 2013 - Short Papers*. Ed. by M.-A. Otaduy and O. Sorkine. The Eurographics Association. DOI: 10.2312/conf/EG2013/short/081-084. URL: http://lgg.epfl.ch/publications/2013/lego/.

Tezuka, Takehito and Hiroo Takada (1992). "Robot Oriented Modular Construction System". In: *9th ISARC, The 9th International Symposium on Automation and Robotics in Construction*, pp. 123–133.

The Assignment Problem and the Hungarian Method. URL: http://www.math.harvard.edu/archive/20_spring_05/handouts/assignment_overheads.pdf.

The Hungarian algorithm: An example. URL: http://www.hungarianalgorithm.com/examplehungarianalgorithm.php.

The Hungarian algorithm: An example. URL: http://csclab.murraystate.edu/~bob.pilgrim/445/munkres.html.

Tibbits, Skylar (2012). "Design to self-assembly". In: *Architectural Design* 82.2, pp. 68–73. ISSN: 00038504. DOI: 10.1002/ad.1381.

Tibbits, Skylar and Kenny Cheung (2012). "Programmable materials for architectural assembly and automation". In: *Assembly Automation* 32.3, pp. 216–225. ISSN: 0144-5154. DOI: 10.1108/01445151211244348. URL: http://www.emeraldinsight.com/doi/10.1108/01445151211244348.

Tirpak, Thomas M (2000). "Design-to-manufacturing information management for electronics assembly". In: *International Journal of Flexible Manufacturing Systems* 12.2-3, pp. 189–205.

Truss, John (1998). *Discrete mathematics for computer scientists*. Addison-Wesley Longman Publishing Co., Inc.

Vigneron, Antoine (2014). "Geometric optimization and sums of algebraic functions". In: *ACM Transactions on Algorithms (TALG)* 10.1, p. 4. DOI: 10.1145/2532647.

Wang, Liyu, Mark M. Plecnik, and Ronald S. Fearing (2016). "Robotic folding of 2D and 3D structures from a ribbon". In: *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June, pp. 3655–3660. ISSN: 10504729. DOI: 10.1109/ICRA.2016.7487550.

Wawerla, Jens, Gaurav S Sukhatme, and Maja J Mataric (2002). "Collective construction with multiple robots". In: *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*. Vol. 3. IEEE, pp. 2696–2701.

Werfel, Justin, Kirstin Petersen, and Radhika Nagpal (2014). "Designing collective behavior in a termite-inspired robot construction team". In: *Science* 343.6172, pp. 754–758. DOI: 10.1126/science.1245842.

Werfel, Justin K, Kirsten Petersen, and Radhika Nagpal (2011). "Distributed multi-robot algorithms for the TERMES 3D collective construction system". In: Institute of Electrical and Electronics Engineers.

Willmann, Jan et al. (2012). "Aerial robotic construction towards a new field of architectural research". In: *International journal of architectural computing* 10.3, pp. 439–460. DOI: 10.1260/1478-0771.10.3.439.

Wilson, Robin J and John J Watkins (1990). *Graphs: an introductory approach: a first course in discrete mathematics*. John Wiley & Sons Inc.

Wismer, Stefan et al. (2012). "Autonomous construction of a roofed structure: Synthesizing planning and stigmergy on a mobile robot". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 5436–5437.

Worcester, James, Joshua Rogoff, and M Ani Hsieh (2011). "Constrained task partitioning for distributed assembly". In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, pp. 4790–4796. DOI: 10.1109/IROS.2011.6095046.

Wurman, Peter R and Joseph M Romano (2015). "The amazonpicking challenge 2015". In: *IEEE Robotics and Automation Magazine* 22.3, pp. 10–12.

Xie, Kai, Jie Yang, and Yue Min Zhu (2007). "Fast collision detection based on nose augmentation virtual surgery". In: *computer methods and programs in biomedicine* 88.1, pp. 1–7.

Xu, Zhe, Connor Mccann, and Aaron M Dollar (2016). "Design of a Reconfigurable Modular Chain for Folding 3D Lattice Structures". In: pp. 1–7. ISSN: 1942-4302. DOI: 10.1115/DETC2016-59496.

Xu, Zhe, Connor McCann, and Aaron M. Dollar (2017). "Reconfigurable Modular Chain: A Reversible Material for Folding Three-Dimensional Lattice Structures". In: *Journal of Mechanisms and Robotics* 9.2, p. 025002. ISSN: 1942-4302. DOI: 10.1115/1.4035863. URL: http://mechanismsrobotics.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4035863.

Yim, Mark (1994). "Locomotion with a unit-modular reconfigurable robot". PhD thesis. stanford university.

Yim, Mark et al. (2007). "Modular self-reconfigurable robot systems: grand challenges of robotics". In: *IEEE Robotics & Automation Magazine* 14.1, pp. 43–52. DOI: 10.1109/MRA.2007.339623.

Yim, Mark et al. (2009). "Modular self-reconfigurable robots". In: *Encyclopedia of complexity and systems science*. Springer, pp. 5618–5631.

Young, Michael (2017a). *Lasvit: CLOVER*. http://www.archiproducts.com/en/products/lasvit/led-handmade-blown-glass-pendant-lamp-clover-led-pendant-lamp_157490.

Young, Michael (2017b). *Lasvit: SUPERCLOVER*. http://www.archiproducts.com/en/products/lasvit/superclover_101494.

Yu, Seung-Nam et al. (2009). "Feasibility verification of brick-laying robot using manipulation trajectory and the laying pattern optimization". In: *Automation in Construction* 18.5, pp. 644–655. DOI: 10.1016/j.autcon.2008.12.008.

Zal (2017). *grabcraft: Feudal Japanese Osaka Castle*. http://www.grabcraft.com/minecraft/feudal-japanese-osaka-castle/castles.

Zeng, Andy et al. (2017). "Robotic Pick-and-Place of Novel Objects in Clutter with Multi-Affordance Grasping and Cross-Domain Image Matching". In: *arXiv preprint arXiv:1710.01330*.

Zhang, Cha and Tsuhan Chen (2001). "Efficient feature extraction for 2D/3D objects in mesh representation". In: *Image Processing, 2001. Proceedings. 2001 International Conference on*. Vol. 3. IEEE, pp. 935–938.

Zhang, Man et al. (2016). "Component-based building instructions for block assembly". In: *Computer-Aided Design and Applications*, pp. 1–8. DOI: 10.1080/16864360.2016.1240450.