UNIVERSITY OF CALGARY

Designing Self-Assembling Systems Via Physically Encoded Information

by

Navneet Bhalla

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

December, 2011

Abstract

Throughout nature, complex self-assembled entities emerge from decentralized components governed by simple rules. Natural self-assembly is dictated by components, their environment, and the interactions among them – physical information – forming a system, describable by a set of rules. The process of self-assembly is equivalent to performing a physical computation, using the interaction and transformation of physically encoded information in a system to build physical information structures – entities as the output. However, designing artificial self-assembling systems remains an elusive goal. Understanding the interplay between information and the generation of a process is required for engineering emergence in this form of natural computing. To investigate this interplay, a self-assembly design methodology was developed, the *three-level approach*, comprising of: (1) specifying a set of rules, (2) modelling these rules to determine the outcome of a system in software, and (3) translating to a physical system by mapping the set of rules using physically encoded information. The contributions of this thesis stem from using the three-level approach to demonstrate how physical information can be used to: enable the self-assembly of desired entities while reducing errors during the process, address the algorithmic constraints of the problem of designing self-assembling systems, and divide the self-assembly process into time-intervals to create more complex desired entities not otherwise possible. These contributions are substantiated through a set of proof-of-concept experiments. Mechanical components in two and three spatial dimensions (in terms of component movement), are confined to a surface or suspended in a fluid. Vibrational energy from the environment facilitates component mobility. Component shape and magnetic-bit patterns are used to encode interactions. Components and environments are fabricated using rapid prototyping. The successful results demonstrate the feasibility of designing self-assembling systems via physically encoded information.

Acknowledgements

This thesis would not have been possible without the support and guidance from many people. I would like to take this opportunity to express my gratitude to all the people involved in the journey that was my doctoral thesis. First, I would like to thank my family and friends for their endless support throughout my Ph.D.

During my Ph.D. I had the opportunity to work with people at several universities. I would like to thank the members of the Evolutionary and Swarm Design lab and the members of the Vize lab at the University of Calgary, the members of the nexus for University College London Evolutionary Algorithms Research (nUCLEAR) group at University College London, and members of the Institut de Recherches Intedisciplinaires et de Dévelopments en Intelligence Artificielle (IRIDIA) at the Université Libre de Bruxelles for engaging in interesting discussions over the years.

I would like to thank Charanjit Bilkhu and Luc Martin at Surtek Precision Machining Inc., Gary Campbell and his team at Nova Product Development Services Ltd., and Craig LeBlanc and Christopher Massie at the Faculty of Environmental Design at the University of Calgary for their assistance in fabricating parts for my experiments.

I would like to thank the members of my supervisory committee: Dr. Christian Jacob, Dr. Peter J. Bentley, Dr. Jeffrey E. Boyd, and Mr. Gerald M. Hushlak. In particular, I would like to thank Jerry for allowing me to collaborate with him on a number of art projects.

I would like to thank Dr. Michael Richter and Dr. Metin Sitti for their guidance. I would like to thank Dr. Anders Nygren for giving me an old orbital shaker, so that I could start my experiments. I am tremendously grateful to Dr. Stuart Kauffman for supporting my research and for his insightful feedback.

A very special thank you to Dr. Marco Dorigo for hosting me at IRIDIA as a visiting

scientist. During my time there Marco presented me with a number of unique opportunities, and his guidance has helped shape my research. I have had the privilege to visit IRIDIA several times after, and Marco's advice has been truly helpful.

I would not have been able to complete my experiments without the help of Dr. Peter D. Vize. Thank you so much Peter for letting me use your lab and for helping me solve some problems with my preliminary experiments.

I have had the privilege to work with Dr. Peter J. Bentley starting with my M.Sc. at University College London and throughout my Ph.D. (including as a visiting scientist). I will be forever grateful to you for encouraging me to get started in this fascinating area of research, and for all your help in making me a better scientist.

Lastly, I would like to thank my supervisor, Dr. Christian Jacob. Thank you Christian for allowing me to collaborate with a number of researchers and pursue my scientific goals.

Table of Contents

Abstract ii
Acknowledgements iii
Table of Contents v
List of Tables viii
List of Figures xi
1 Introduction 1
1.1 Scientific Motivation 4
1.2 Thesis Hypothesis 9
1.3 Thesis Objectives 11
1.4 Thesis Structure 12
2 Literature Review 15
2.1 Physical Self-Assembling Systems 16
2.1.1 Macroscale Self-Assembly 17
2.1.2 Mesoscale Self-Assembly 25
2.1.3 Nanoscale Self-Assembly 30
2.2 Physical Information Encoding and DNA Computing 33
2.2.1 Physical Information Encoding in Nature 33
2.2.2 DNA Computing 35
2.2.3 The abstract Tile Assembly Model 35
2.2.4 Extensions to the abstract Tile Assembly Model 38
2.2.5 The Complexity of Self-Assembly 40
2.3 Designing Self-Assembling Systems 46
2.3.1 Top-Down Self-Assembly Design 46
2.3.2 Bottom-Up Self-Assembly Design 47
2.4 Summary 49
3 Physical Information Analysis 52
3.1 An Idealized Self-Assembly Model 53
3.1.1 Framework 53
3.1.2 Idealized Model Design 56
3.1.3 Prototypes 62
3.1.4 Prototype Results 66
3.2 A Self-Reproducing Analogue 70
3.2.1 System Overview 70
3.2.2 Rules 71
3.2.3 Specifications 74
3.2.4 Test Results 77
3.3 Summary 78
4 The Three-Level Approach to Self-Assembly Design 82
4.1 Level One: Definition of Rule Set 84
4.2 Level Two: Virtual Execution of Rule Set 86
4.3 Level Three: Physical Realization of Rule Set 90

4.3.1 Component Design Space 92
4.3.2 2D Physical Information Encoding Schemes 93
4.3.3 3D Physical Information Encoding Schemes 98
4.4 Summary 102
5 Programming Self-Assembling Systems 104
5.1 Hypothesis Statement 104
5.2 2D Programming Experiments and Results 105
5.2.1 Level One: Rule Set for 2DP Experiments 106
5.2.2 Level Two: Virtual Execution of Rule Set for 2DP Experiments . 107
5.2.3 Level Three: Physical Realization of Rule Set for 2DP Experiments 108
5.3 3D Programming Experiments and Results 113
5.3.1 Level One: Rule Set for 3DP Experiments 114
5.3.2 Level Two: Virtual Execution of Rule Set for 3DP Experiments . 116
5.3.3 Level Three: Physical Realization of Rule Set for 3DP Experiments 117
5.4 Summary 122
6 Evolving Self-Assembly Rule Sets 123
6.1 Evolutionary Algorithm 123
6.1.1 Genotype and Phenotype Representations 125
6.1.2 Multi-Objective Fitness Function 126
6.1.3 Selection, Crossover, and Genetic Operators 132
6.2 Hypothesis Statement 133
6.3 2D Evolutionary Experiments and Results 134
6.3.1 Level One: Rule Set for 2DE Experiments 134
6.3.2 Level Two: Virtual Execution of Rule Set for 2DE Experiments . 135
6.3.3 Level Three: Physical Realization of Rule Set for 2DE Experiments 137
6.4 3D Evolutionary Experiments and Results 139
6.4.1 Level One: Rule Set for 3DE Experiments 140
6.4.2 Level Two: Virtual Execution of Rule Set for 3DE Experiments . 140
6.4.3 Level Three: Physical Realization of Rule Set for 3DE Experiments 142
6.5 Summary 144
7 Staging the Self-Assembly Process 147
7.1 Staging and the Three-Level Approach 149
7.1.1 Staging - Level One: Definition of Rule Set 149
7.1.2 Staging - Level Two: Virtual Execution of Rule Set 150
7.1.3 Staging - Level Three: Physical Realization of Rule Set 151
7.2 Hypothesis Statement 152
7.3 2D Staging Experiments and Results 153
7.3.1 Level One: Rule Set for 2DS Experiment 154
7.3.2 Level Two: Virtual Execution of Rule Set for 2DS Experiment . . 154
7.3.3 Level Three: Physical Realization of Rule Set for 2DS Experiment 155
7.4 3D Staging Experiments and Results 157
7.4.1 Level One: Rule Set for 3DS Experiments 159
7.4.2 Level Two: Virtual Execution of Rule Set for 3DS Experiments . 159
7.4.3 Level Three: Physical Realization of Rule Set for 3DS Experiments 160

7.5 Summary 162
8 Discussion 166
8.1 Summary and Analysis of the Experimental Results 166
8.2 Leveraging Limited Rule Sets 174
8.3 Future Work 180
8.3.1 Alternative Physical Information Encoding Schemes 180
8.3.2 Broad Directions 184
8.3.3 Error Correction 186
8.3.4 Scaling 187
8.4 Applications 188
8.5 Summary 189
9 Conclusions 191
9.1 Thesis Summary 191
9.2 Thesis Hypothesis Satisfaction 196
9.3 Thesis Contributions 198
9.4 Thesis Publications 199
9.5 Thesis Conclusions 201
Bibliography 202
A Component Specifications 224
A.1 Penrose Components 224
A.2 2D Components 224
A.3 3D Components 229
B Environment Specifications 234
B.1 Penrose Environment 234
B.2 2D Environment 236
B.3 3D Environment 239
C Experimental Procedures 243
C.1 Experimental Procedure for the Penrose Systems 243
C.2 Experimental Procedure for the 2D Systems 244
C.3 Experimental Procedure for the 3D Systems 245

List of Tables

3.1 Idealized model summary 63
3.2 Parameter settings and target structure features for the five prototypes (P): number of simulation components (#SC), component velocity (CV), environment radius (ER), rule type (RT) of either global axes (GA) and/or axes count (AC) information, number of rules (#R), number of components (#C), and symmetric/asymmetric (S/A) features 67
3.3 Comparison of the features of the idealized model and the original Penrose self-reproducing analogue 80

4.1 2D interaction rules (fits and breaks; '→' transition, '+' assembly, ';' disassembly, and 'ϕ_2' temperature 2), where A-H represent component information. 87
4.2 3D interaction rules (fits and breaks; '→' transition, '+' assembly, ';' disassembly, and 'ϕ_2' temperature 2), where I-T represent component information and 360, 180, and 90 represent component rotational interactions. 87
4.3 Model features of the cTAM compared to the aTAM 88
4.4 First key and lock designations to the 3-magnetic-bit patterns (red/zero and blue/one represent magnetic south and north respectively) 94
4.5 Second key and lock designations to the 3-magnetic-bit patterns (red/zero and blue/one represent magnetic south and north respectively) 97
4.6 First key and lock designations to the 5-magnetic-bit patterns (red/zero and blue/one represent magnetic south and north respectively) 100
4.7 Second key and lock designations to the 5-magnetic-bit patterns (red/zero and blue/one represent magnetic south and north respectively) 101

5.1 Programmed and random (P/R) component sets (represented as '# × (Top, Left, Bottom, Right)', where # represents quantity and the directions refer to component information) for the 2DP experiments (EX) . . 107
5.2 The number of successful and unsuccessful trials for each programmed and random 2DP experiment (2DPE1 - 2DPE3) at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data 109
5.3 The number of successful and unsuccessful trials for the 2DP experiments at level three, with corresponding p-values (calculated using Fisher's Exact Test (one-sided) for analyzing binary data with respect to 0 successful and 10 unsuccessful trials for the random component sets for each 2DP experiment from their level two results) 113
5.4 Programmed and random (P/R) components sets (represented as '# × (Top, Left, Bottom, Right, Front, Back)', where # represents quantity and the directions refer to component information) for the 3DP experiments (EX) 115

5.5 The number of successful and unsuccessful trials for each programmed and random 2DP experiment (2DPE1 - 2DPE3) at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data 117

5.6 Number of successful and unsuccessful trials for the programming experiments (PEX1 - PEX5) at level three, with corresponding p-values (calculated using Fisher's Exact Test (one-sided) for analyzing binary data, with respect to 0 successful and 10 unsuccessful trials for the random sets for each programming experiment from their level two results) 119

6.1 Evolutionary algorithm parameter settings for 2D and 3D systems. . . . 125

6.2 A sliding window technique is used (matrix) as a sum of magnetic errors (odd number of magnets must match at each position along the sliding window) and is applied to all potential two-component key-to-key interactions in a system 127

6.3 Magnetic error interactions matrix (in reference to Table 4.7, page 101), where the numbers are the sum of all errors between information in the four orientations (using the number of magnets, i.e. two or one, in each mismatched location), and where N/A is in reference to lock-to-lock interactions not being possible 131

6.4 Evolved and random (E/R) component sets (represented as '# × (Top, Left, Bottom, Right)', where # represents quantity and the directions refer to component information) for the 2DE experiments (EX) 136

6.5 The number of successful and unsuccessful trials for each evolved and random evolutionary experiment (2DEE1 and 2DE2) at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data 137

6.6 Number of successful and unsuccessful trials for the evolutionary experiments (2DEE1 and 2DEE2) at level three, with corresponding p-values (calculated using Fisher's Exact Test (one-sided) for analyzing binary data, with respect to 0 successful and 10 unsuccessful trials for the random sets for each evolutionary experiment from there level two results) 140

6.7 Evolved and random (E/R) components sets (represented as '# × (Top, Left, Bottom, Right, Front, Back)', where # represents quantity and the directions refer to component information) for the 3DE experiments (EX) 141

6.8 The number of successful and unsuccessful trials for each evolved and random evolutionary experiment (3DEE1 - 3DEE3) at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data 143

6.9 Number of successful and unsuccessful trials for the evolutionary experiments (3DEE1 - 3DEE3) at level three, with corresponding p-values (calculated using Fisher's Exact Test (one-sided) for analyzing binary data, with respect to 0 successful and 10 unsuccessful trials for the random sets for each evolutionary experiment from there level two results) 145

7.1 Staged component set (represented as '*psi* # × (Top, Left, Bottom, Right)', where ψ identifies the time interval, # represents quantity, and the directions refer to component information) for the 2DS experiment (EX) . . . 155

7.2 The number of successful and unsuccessful trials for the staged and non-staged trials for 2DSE1 at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data 156

7.3 The number of successful and unsuccessful trials for the staged and non-staged trials for 2DSE1 at level three, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data . . 157

7.4 Staged component set (represented as '*psi* # × (Top, Left, Bottom, Right, Front, Back)', where ψ identifies the time interval, # represents quantity, and the directions refer to component information) for the 3DS experiment (EX) 160

7.5 The number of successful and unsuccessful trials for the staged and non-staged trials for 3DSE1 - 3DSE3 at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data 161

7.6 Number of successful and unsuccessful trials for the staging experiments (3DSE1 - 3DSE3) at level three, with corresponding p-values calculated using Fisher's Exact Test (one-sided) for analyzing binary data 162

8.1 Summary of the level two experiments, showing the number of successful and unsuccessful trials for the designed component groups of the programming (2DP and 3DP), evolving (2DE and 3DE), and staging (2DS and 3DS) experiments, and corresponding p-values calculated using Fisher's Exact Test (one-sided) for analyzing binary data where all the control component groups had zero successful and ten unsuccessful trials 169

8.2 Summary of the level three experiments, showing the number of successful and unsuccessful trials for the designed component groups of the programming (2DP and 3DP), evolving (2DE and 3DE), and staging (2DS and 3DS) experiments, and corresponding p-values calculated using Fisher's Exact Test (one-sided) for analyzing binary data where all the control component groups had zero successful and ten unsuccessful trials 171

8.3 5-magnetic-bit encoding scheme to create the dog target structure, where the differences between the encoding scheme used for the previous 3D experiments (Table 4.7 page 100) are highlighted 178

8.4 The number of self-matching errors using a 9-magnetic-bit pattern and the corresponding number of pairs for each category of rotational information, where ROT-1 - ROT-4 correspond to a 90° planar rotation between the faces of two components with the same rotational information 183

List of Figures

1.1 What are the requirements to have a set of parts self-assemble into a desired entity? . . . 1

1.2 Resulting self-assembled snow crystal morphologies depending on the temperature and the supersaturation level of water vapour (Libbercht 2001). 3

1.3 Components comprising the bacterial flagellum, and their self-assembly sequence. . . . 4

2.1 Mechanical macroscale self-assembling systems: (I) extension to the original self-reproducing system (Penrose 1958), (II) hexagon target structure (Hosokawa et al. 1996), (III) using component shape and magnetic patterns to create target structures with symmetric/asymmetric features (Bhalla & Bentley 2006), (IV) open target structures (Miyashita et al. 2009), (V) self-replication using an oscillating environment (Breivik 2001), (VI) self-reconfiguration (Reprinted (adapted) with permission from Mao et al. 2002, © 2002 American Chemical Society), (VII) cube components directed using the environment (Kalontarov et al. 2010), and (VIII) virus model (Olson et al. 2007). . . . 18

2.2 Robotic macroscale self-assembling systems: (I) 2D self-replication (Griffith et al. 2005), (II) reprogrammable components (Bishop et al. 2005, © 2005 IEEE), (III) folding sheet (Hawkes et al. 2010), (IV) 3D templated self-assembly (White et al. 2005), (V) Claytronics self-reconfigurable robot (Goldstein et al. 2005), and (VI) Swarm-bot (Mondada et al. 2003, © 2003 IEEE). . . . 23

2.3 2D mesoscale self-assembling systems: (I) triangular and circular components (Reprinted (adapted) with permission from Choi et al. 2000, © 2000 American Chemical Society), (II) open structure with deliberate hexagonal holes (Bowden et al. 1997), and (III) aperiodic tiling (Rothemund 2000). 26

2.4 3D mesoscale self-assembling systems: (I) cylinder using key-lock shape (Terfort et al. 1997), (II) open chiral structure (Reprinted (adapted) with permission Boncheva et al. 2003a, © 2003 American Chemical Society), (III) staged self-assembly using spherical components (Reprinted (adapted) with permission Wu et al. 2002, © 2002 American Chemical Society), (IV) cube with patterned faces (Garcias et al. 2002), (V) 3D circuit (Garcias et al. 2000), and (VI) casquet made using micro-masonry (Fernandez & Khademhosseini 2010). . . . 29

2.5 DNA nanoscale self-assembling systems: (I) truncated-octahedron (Reprinted (adapted) with permission Zhang & Seeman 1994, © 1994 American Chemical Society), (II) DNA Tiles (Winfree et al. 1998a), (III) 2D DNA origami (Rothemund 2006), (IV) 3D DNA origami (Douglas et al. 2009), (V) 2D DNA origami using shapes (Woo & Rothemund 2011), and (VI) staged tetrahedron (He et al. 2008). . . . 31

2.6 A set of 13 Wang tiles that only result in an aperiodic tiling of the plane. 36
2.7 An example of the benefit of reducing the amount of information required when rotation is permitted (bottom) compared to when rotation is not permitted (top) between components with complementary information (represented using colours, similar to Wang tiles where tiles with the same coloured edges can be assembled). 45

3.1 2D square lattice pattern (left) and diamond lattice pattern (right). . . . 58
3.2 Global axes information for 2D square lattice pattern, with component (0,0) positioned at the centre of the grid. 60
3.3 Axes count information (original configuration, left), with case one indicated with the black-outlined white-filed component, and case two indicated with the black-outlined grey-filled component. 61
3.4 The two black-outlined components with the same axes count information (left); and the same two with unique global axes information (right). . . . 62
3.5 Four target structures: cube (top-left), chair (top-right), NB (bottom-left), and mug (bottom-right); © 2007 IEEE. 63
3.6 3D cubic lattice (left), 2D hexagonal lattice (centre), and 3D layered hexagonal lattice (right); © 2007 IEEE. 64
3.7 Sticky site locations for the 3D hexagonal grid, with positions 6 and 7 located at the centre-front and centre-back of the component indicated by the black circle. 65
3.8 Global axes information for a 2D hexagonal lattice shown as a 2D square grid (left), and the global axes information values (right). 66
3.9 Five prototype results; © 2007 IEEE. 68
3.10 Four examples of different mug configurations, demonstrating the bottom-up parallel construction characteristic of self-assembly; © 2007 IEEE. . . 69
3.11 From left to right: component α and β (in their neutral positions), and complex $\alpha\beta$ and $\beta\alpha$. 71
3.12 Example scenario with initial configuration (top) and final configuration (bottom). 71
3.13 α (left) and β (right) identifying regions of physically encoded information using shapes. 72
3.14 $\alpha\beta$ assembly (left, using rule (3.1)) and $\beta\alpha$ assembly (right, using rule (3.2)). 72
3.15 Rotation rules (3.3) to (3.6) promote the creation of new $\alpha\beta$ complexes. . 73
3.16 Rotation rules (3.7) to (3.10) promote the creation of new $\beta\alpha$ complexes. 73
3.17 Environment height constraints (represented with lines) prevents components from flipping (left), and provides rotation flexibility to aid in bond creation between components (right). 74
3.18 Machined component α (left) and β (right). 76
3.19 Initial (left) and final (right) configurations after running each of the four system tests. 77

4.1 The three-level approach to self-assembly design. 83

4.2 2D and 3D component spatial information relationship (I and V respectively), an example of information orientation on a 3D component's face (VII), and example 2D component types (where III and IV are of the same type under planar rotation and are distinguished from II). 86
4.3 2DcTAM example steps. 90
4.4 Examples of assembly violations (left) and no assembly path (right). . . . 91
4.5 Design space for the first 2D physical information encoding scheme (left), and an example component showing the shortcoming of neighbouring lock shapes (right). 93
4.6 Self-assembling systems using the first 2D physical information encoding scheme (Bhalla et al. 2007). 95
4.7 Design space for the second 2D physical information encoding scheme (left), and an example component with information markers. 96
4.8 Six pairs of unique 5-magnetic-bit patterns, where blue and red represent magnetic north and south respectively, along with rotational properties (360°,180°, and 90°), and colours associated with each pair (used as paint markers on components to indicate information associated with component-to-component interactions). 99
4.9 3D component shape space (left; showing the base component with information markers, the minimum component with lock shapes on all six faces, and the maximum component with key shapes on all six faces), and example components (right). 99

5.1 From left to right: target structures for the 2DP experiments corresponding to 2DPE1 to 2DPE3 respectively. 105
5.2 Number of target structures created at the end of each trial (TR1 - TR10), for each 2DP experiment (2DPE1 - 2DP3). 110
5.3 Photographs of successful 2DP experiments (2DPE1 - 2DPE3) and an example error (trial from 2DPE3), where adjacent components to the outlined shapes are not attached to the self-assembled structures (resting positions when energy was removed from the environment). 111
5.4 Number of matching errors at the end of each trial (TR1 - TR10), for each 2DP experiment (2DPE1 - 2DP3). 112
5.5 From left to right: target structures for the 3DP experiments corresponding to 3DPE1 to 3DPE5 respectively (with perspective, top, and front views). 114
5.6 Number of target structures created in each of the ten trials (TR1 - TR10), for each of the five programming experiments (3DPE1 - 3DPE5) at level three. 119
5.7 Photographs of successful level three programming experiment trials (PEX1 - PEX5). 120
5.8 Number of rotation errors in each of the ten trials (TR1 - TR10), for each of the five programming experiments (PEX1 - PEX5) at level three. . . 121

6.1 Three-level approach (left) incorporating evolutionary computing (right). 124
6.2 Example of a 3D genotype that has more than one phenotype, where the 180° bond between information M and N can assemble in two different configurations (shown with the dashed lines). 126
6.3 Fitness objective examples: structure I ($A = 5$, $P = 12$, and $E = 1$); structure II has the same second moment of inertia for its reflected equivalent; number of matches between reflected structure II is 3 (III); number of open locations is 2 (black circles, IV); an error of 2 in IV (Table 6.2). . 128
6.4 Fitness objective examples: structure I ($V = 3$, $S = 14$, $B = 2.5$, and $E = 1$), structures II and III have the same moment of inertia, the number of matches between structures II and III which have similar reflected features is 3, the number of open locations is 2 (indicated using black circles in IV; 2D top view), and the potential error score in IV is 8 (Table 6.3). 130
6.5 Target structures for 2DEE1 (left) and 2DEE2 (right). 134
6.6 The two optimal solutions for 2DEE1 (I and II) and the single optimal solution for 2DEE2 (III). 135
6.7 Number of target structures created in each of the ten trials (TR1 - TR10), for each of the two evolutionary experiments (2DEE1 and 2DEE2) at level three. 138
6.8 Photographs of successful 2DE experiments (2DEE1 and 2DEE2), where adjacent components to the outlined shapes are not attached to the self-assembled structures (resting positions when energy was removed from the environment). 139
6.9 From left to right: target structures for the 3DE experiments corresponding to 3DEE1 to 3DEE3 respectively (with perspective, top, and front views). 141
6.10 Number of target structures created in each of the ten trials (TR1 - TR10), for each of the three evolutionary experiments (3DEE1 - 3DEE3) at level three. 144
6.11 Photographs of successful level three evolutionary experiment trials (3DEE1 - 3DEE3). 145

7.1 2DscTAM example where two time intervals are used to self-assemble a 3×3 square target structure. 151
7.2 2D staged target structure for the 2DS experiment (2DSE1). 153
7.3 Staging strategy for the 2D target structure, and error prevention due to shape and proper 3-magnetic-bit pattern selection (e.g. avoid magnetic repulsion configuration). 154
7.4 Photographs of the successful 2DS experiment (2DSE1) after the first time interval (left) and the second time interval (right). 157
7.5 From left to right: target structures for the 3DS experiments corresponding to 3DSE1 to 3DSE3 respectively (with perspective, top, and front views). 158
7.6 Staging strategy for the second 3D target structure (applicable to the first and third target structure). 159

7.7 Photographs of successful level three staged experiment trials (3DSE1 - 3DSE3) after the first time interval (left) and the second time interval (right). 163
7.8 Rotational errors at the end of each 3DS staged trial. 164
7.9 Rotational errors at the end of each 3DS non-staged trial. 165

8.1 Four views of the dog target structure. 175
8.2 The dog target structure showing an assembly between neighbouring components (solid black circle) and the 90° rotational requirements (dashed lines), where the grouped connections can use the same rotational information due to symmetry (and the four corner components in the body can use the same rotational information, top view). 177
8.3 Staging for a single dog target structure, where colours are used to identify the six time intervals. 179
8.4 Staging for multiple dog target structures, where colours are used to identify the six time intervals (the last three time intervals correspond to the last three time intervals used to create a single dog target structure Figure 8.3 page 179). 181

A.1 Mathematical specifications for the Penrose components α (left) and β (right). 225
A.2 2D magnet tool specification (top/bottom and front view) - all magnet holes on the sides have the same radius (depth at key sites is 1.35 millimetres, and depth at lock sites is 2.20 millimetres), and all paint holes on the top an bottom have the same radius (all construction units in millimetres). 227
A.3 2D component protection tool (top view) - the height of this part is 5 millimetres (all construction units in millimetres). 227
A.4 2D component specification (top view and front view) - all magnet holes on the sides have the same radius, and all paint holes on the top have the same radius and depth (all construction units in millimetres). 228
A.5 Example of constructing a 2D component during a vice press, and 2D magnet placement tool (left) and 2D component (right) inside the vice. . 228
A.6 3D magnet placement tool specifications (counter-clockwise from top right: perspective, top, front, right views), with all construction units in millimetres; note two magnet locations are shown at the top of the component (centre and corner), however only one magnet location should be used per tool (two locations are show here to reduce the number of technical drawings). 231
A.7 3D Component specifications showing the base component dimensions, key shape, lock shape, and interaction markers (counter-clockwise from top right: perspective, top, front, right views), with all construction units in millimetres. 232
A.8 Example of constructing a 3D component during a vice press, and 3D magnet placement tool (left) and 3D component (right) inside the vice. . 233

B.1 Specification for the three parts, front and back parts (top) and centre part (bottom), making the Penrose environment (all construction units in millimetres; note hole diameter is 4 millimetres). 235
B.2 Fully constructed physical Penrose environment. 235
B.3 Tray base specification (top view) - the outer radius is 135; the inner radius is 125; the outer wall height is 9; the inner wall height is 6; the radius for each screw hole is 2 (all construction units in millimetres). 237
B.4 Tray mounting bracket specification (surface specification, top view) - the outer radius is 220; in each group of five screw holes, the distance from closest to furthest (centre of the hole to the centre of the part) is 55, 67.5, 80, 92.5, 105; the radius for each screw holes is 2 (all construction units in millimetres). 238
B.5 Tray mounting bracket specification (mount detail, front view) - smaller bracket surface for clarity (all construction units in millimetres). 238
B.6 Tray lid specification (top view) - the radius is 135; the height is 2; the radius for each screw hole is 2 (all construction units in millimetres). . . 239
B.7 From left to right: steps five to eight to construct the 2D environment. . 239
B.8 Environment parts for jar rack (from top left to bottom: bottom part X, bottom part Y, top part, jar sleeve, and side part), with all construction units in millimetres. 241
B.9 Photograph of the 3D environment with the jar rack secured to the shaker, and a jar with mineral oil containing components as an example (jars of mineral oil are used to finalize the 3D components environment; details are provided in *section C.3: Experimental Procedures for the 3D Systems*). 242

C.1 Jar configurations (top view) for the 3DP experiments (left, 3DPE1 - 3DPE5), the 3DE experiments (centre, 3DEE1 - 3DEE3), and the 3DS experiments (right, 3DSE1 - 3DSE3). 246

Chapter 1

Introduction

Place the individual pieces of a jigsaw puzzle in a box and shake. Could a jigsaw puzzle effectively construct itself resulting in a fixed structure representing the solution (Figure 1.1)? Or, put the parts of a car in an even larger box and shake. Would this action result in a complex machine, a car? These two metaphors were used in the past to depict what could be achieved through the mastery of self-assembly to compute solutions and create devices (Jones 2004; Drexler 1986). Self-assembly, the autonomous construction of a set of building blocks in an environment into a desired entity, is not merely in the realm of science fiction. From granules of salt to galaxies and from bacteria to blue whales, the plethora of complex inorganic and organic systems seen throughout nature are the result of self-assembly. Self-assembly is fundamental to a variety of natural processes, including crystalline growth and biological development.

The old adage that no two snow flakes are alike has been demystified by understanding the self-assembly process of snow crystals. Linus Pauling was the first to propose a description to the molecular structure of the ice crystal (1935). Quantum mechanics explains how two hydrogen atoms and one oxygen atom form a single water molecule, and

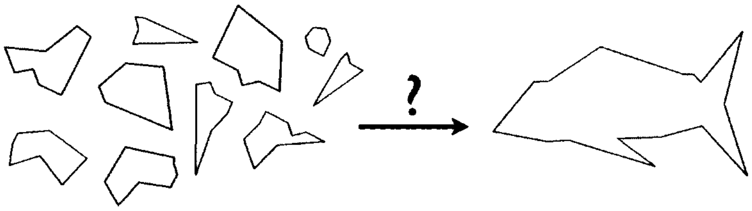


Figure 1.1: What are the requirements to have a set of parts self-assemble into a desired entity?

how multiple water molecules attach together to form a hexagonal crystal lattice (Libbercht 2001). It is this hexagonal symmetry of ice that results in the six-fold symmetry of snow crystals. One differentiating factor to how a snow crystal is formed versus ordinary ice is in the initialization of the process. In nature, the process of creating a snow crystal is initialized through the use of a seed particle, typically a bacterium (Christner et al. 2008) or a dust particle (Libbercht 2001), to which water molecules can attach to. The variation and relation of environment variables leads to the vast range of snow crystal morphologies.

This process has been leveraged to create predictable artificial snow crystals of varying form under a controlled environment. In a laboratory, the self-assembly process is initiated using the tip of an ice needle to act as the seed particle. The ice needle is placed in a chamber, where the supersaturation level of water vapour, temperature, and background gas can be controlled. By understanding the relationship between these three environment variables, it has lead to the predictable self-assembly of artificial snow crystal morphologies (Figure 1.2).

Likewise, investigations into the mechanisms of the self-assembly process is helping to unravel the mysteries of biological development. The outboard motor provides propulsion to boats, and has been used as an analogy to describe the bacterial flagellum that provides one mechanism for bacteria motility (Dawkins 2004). A bacterial flagellum can be considered as a rotary motor, which has a diameter of approximately 30 nanometres, rotates at around 20,000 revolutions per minute (rpm), consumes energy of roughly 10^{-6} Watts, and has energy conversion near 100% (Ishiguro 2004). Unlike an outboard motor, this remarkable machine is created autonomously (Yonekura et al. 2000; Macnab 2003).

Since the tail of the flagellum extends past the bacterial cell wall, many of the flagellum's components are extruded through an export apparatus that self-assembles first, and is part of the base of the final structure. The self-assembly process of the bacte-

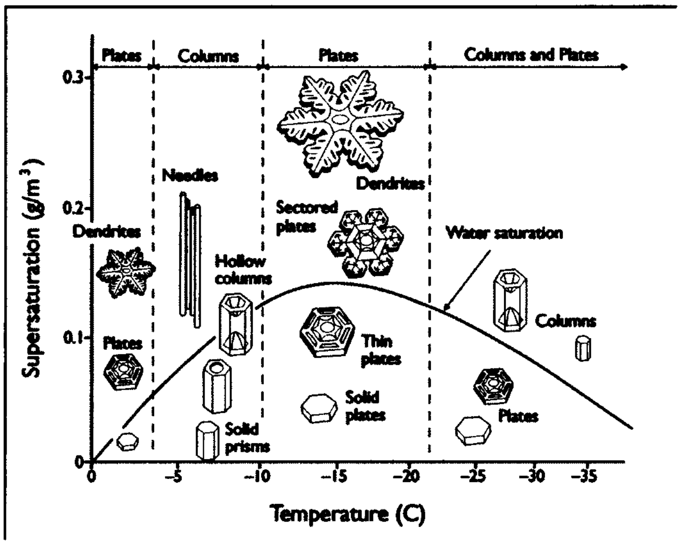


Figure 1.2: Resulting self-assembled snow crystal morphologies depending on the temperature and the supersaturation level of water vapour (Libbercht 2001).

rial flagellum can be viewed as a linear process, where components first form the base (which serves as a mounting plate), followed by the construction of the hook (which acts as a universal joint) second, and finally followed by the formation of the filament third (Figure 1.3). The filament is fragile and suffers from breakage, but can be regrown. The sequence of steps to the self-assembly and self-repair of the bacterial flagellum is understood by the interactions of the flagellum's components.

Snow crystals and bacterial flagellum are just two of the plethora of complex inorganic and organic systems seen throughout nature that are the result of self-assembly. Complex self-assembled entities emerge from decentralized components governed by simple rules. Natural self-assembly is dictated by the morphology of the components and the environmental conditions they are subjected to, as well as their component and environment physical and chemical properties – their information (Ball 1999, 2009a,c,d; Thompson 1917). Components, their environment, and the interactions among them form a system, which can be described by a set of simple rules. One could view entities in nature consti-

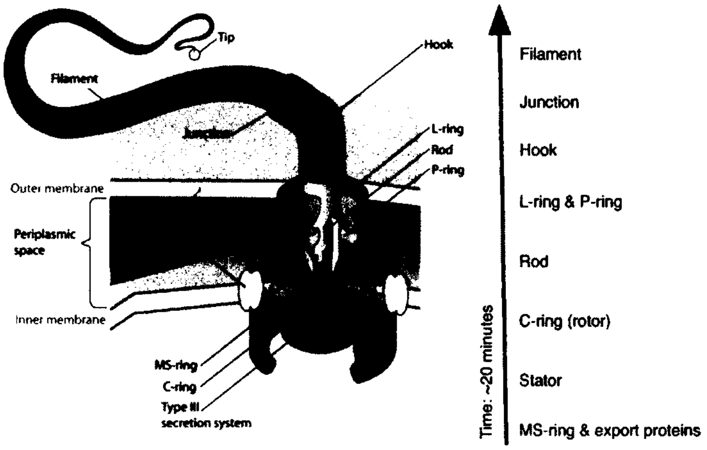


Figure 1.3: Components comprising the bacterial flagellum, and their self-assembly sequence.

tuting programs that are based on the rules present in a system (Wolfram 2002). These programs are the interaction and transformation of physically and chemically encoded information in a system. The process of self-assembly has been shown to being equivalent to performing a physical computation, as information is compared and exploited to build larger structures (Winfree 1995, 1999). However, designing and constructing artificial self-assembling systems remains an elusive goal.

1.1 Scientific Motivation

Comprehending the principles and mechanisms of self-assembly has been described as one of the important aspects to understanding life (Ingber 1998). In addition to this profound goal and understanding natural self-assembly universally, self-assembly is also viewed as an *enabling technology* for the creation of artificial systems (Pelesko 2007). Of the various applications incorporating self-assembly, the most anticipated advantage of self-assembly is in the creation of nanotechnology (Frankel & Whitesides 2009; Whitesides & Grzybowski 2002; Nalwa 2004).

In 1959, Richard Feynman gave is visionary talk (later transcribed; 1960) *There's Plenty of Room at the Bottom*, where he postulated the construction of matter from the molecular level up, atom-by-atom. As part of his vision, Feynman spoke of billions of tiny factories being able to manufacture materials. In a later talk, *Infinitesimal Machinery*, Feynman further developed his vision of molecular machines (with a manuscript written based on his talk; 1993). K. Eric Drexler built upon Feynman's ideas with the notion of self-replicating machines, in *Engines of Creation: The Coming Era of Nanotechnology* (1986). To Feynman and Drexler, the potential impact of nanotechnology was nothing short of creating new materials from the bottom-up, nano-devices delivering medications to target areas in the human body, and nano-machines removing pollutants from the environment (Drexler 1981, 1986, 1992).

Although it is considered as revisionist history to credit Feynman as the founder of nanotechnology (Ball 2009b), and that the details Feynman and Drexler described on achieving their grander visions have been met with controversy (Baum 2003; Whitesides 2001; Committee to Review the National Nanotechnology Initiative & National Research Council 2006), their conceptual visions have nevertheless provided inspiration and have generated interest in nanotechnology (Toumey 2005). The Scanning tunneling microscope (STM; Binnig et al. 1982a,b) is described as the *founding ancestor* to nanotechnology (Ball 2009b; Toumey 2005). The STM, the atomic force microscope (AFM; Binnig et al. 1986), and other technologies (Whitesides & Love 2001; Gates et al. 2005) create nanostructures using traditional top-down design approaches where atoms are placed one-by-one. However, self-assembly is recognized as one of the few practical methods to building ensembles of nanostructures moving forward, as well as being central to dynamic multicomponent systems for the creation of smart materials, self-repairing structures, and self-replicating machines (Whitesides & Grzybowski 2002; Gates et al. 2005).

In addition to these broad motivations for understanding natural self-assembly and developing self-assembly as an enabling technology for the creation of artificial systems, there are several specific motivations to studying self-assembly in the context of this thesis. These specific motivations include:

- creating mesoscale/macroscale (micrometre to millimetre / centimetre and above) self-assembling systems (Whitesides & Boncheva 2002; Boncheva et al. 2003b; Whitesides & Boncheva 2005),
- constructing self-assembling systems in three spatial dimensions (Terfort et al. 1997),
- developing the computational aspects of self-assembly (Stepney et al. 2005, 2006),
- advancing the nascent field of *morphogenetic engineering* (Doursat 2008), and
- investigating complexity theory (Whitesides & Grzybowski 2002).

The physical realization of self-assembling systems is not limited to molecular and nanoscale systems. Both mesoscale and macroscale self-assembly offer a unique quality in that components at these scales offer flexibility in design and access to types of functionality unparalleled in comparison to molecular and nanoscale systems (Whitesides & Boncheva 2005). As a corollary, self-assembly at the millimetre to centimetre scale gives a hands-on approach, which is also a unique opportunity to build knowledge in self-assembly research (Pelesko 2007). However, mesoscale self-assembly has its challenges (Whitesides & Boncheva 2005), which include:

- designing components (e.g. size, shape, and complexity) and environments (e.g. temperature and boundary conditions),
- fabricating components and environments,

- understanding the range of self-assembled structures that are possible in a system and the potential for defects, and
- designing abiological systems.

From a material science perspective, the physical properties for self-assembly in biological systems may not be well suited for systems using non-biological parts. For example, magnetic interactions are not common in biological systems, but the insensitivity of magnetic interactions to environmental influences (such as solvent properties) are regarded as the most promising in mesoscale self-assembly (Whitesides & Boncheva 2005).

From the macroscale to the nanoscale, the creation of self-assembling systems in three spatial dimensions has the potential for a number of advantages. The human brain is one of the greatest examples of three-dimensional self-assembly, with features ranging across these physical scales (Whitesides & Grzybowski 2002). Potential advantages of three-dimensional self-assembly include:

- shorter interconnections between components (Whitesides & Grzybowski 2002),
- efficient use of volume, (Whitesides & Grzybowski 2002), and
- execution of algorithms which would otherwise be more difficult to implement in two spatial dimensions (Pelletier & Weimerskrich 2002).

The advent of three-dimensional self-assembly will push the boundaries of *embodied computation* and *morphogenetic engineering*. Self-assembly is identified as a fundamental aspect to achieving embodied computation (Stepney et al. 2005, 2006), where physical computation is dependent on the underlying substrate for example. Understanding the interplay between programmability/controllability and self-organization is required in order for engineering emergence in this form of natural computing (Li et al. 2008). In addition, understanding this interplay has also been identified for creating new technologies with natural characteristics (e.g. self-repair, self-reproduction, self-reconfiguration, and parallel construction), as is the goal of morphogenetic engineering (Doursat 2008).

Along with practical applications, self-assembly can also be used in theoretical investigations to advance areas such as complexity theory (Mitchell 2006, 2009). It has been argued that self-assembly, by bridging the study of distinct components and the study of many interacting components, can be used to investigate the relationship between reductionism to complexity and emergence (Whitesides & Grzybowski 2002). Self-assembly has been shown to being an algorithmically NP-complete problem (Adleman et al. 2002). The most notable characteristic of an NP-complete problem is that no computationally efficient method is know to solve them; any given solution to an NP-complete problem can be verified efficiently, but there is no efficient method to find a solution to begin with (Sipser 1997). Consequently, it will be more challenging to deploy top-down self-assembly design methodologies as the sophistication of these systems increases.

The development of a design method that inherently combines bottom-up design with bottom-up construction has the potential to advance the theory and practice of self-assembly. For example, instead of a top-down method that takes a desired entity and dissects it into a set of components, a bottom-up method could specify the properties of the components and environmental conditions as well as present what emerges after the system is allowed to run (Wolfram 2002). The development of such a bottom-up self-assembly design method requires the pairing of theory and experiment in order for the science of self-assembly to continue to move forward (Pelesko 2007).

1.2 Thesis Hypothesis

The following hypothesis statement forms the foundation of this thesis.

Thesis Hypothesis: it is possible to encode information as physical components and their corresponding environments to enable the self-assembly of closed structures with desired morphologies.

Where,

Components: are disordered, mobile, fundamental building blocks, which are able to adjust their connectivity with respect to each other (Whitesides & Grzybowski 2002).

Corresponding Environments: are constraints, conditions, and external energy sources for components.

Self-assembly: is an autonomous, bottom-up process that involves the aggregation of components through selective, local interactions in a corresponding environment, where components can equilibrate between aggregated and non-aggregated states (Whitesides & Grzybowski 2002).

Encode Information: to encode information in this work means a process that transforms information into physical information.

Closed Structures: are aggregated components forming an entity with a defined boundary (created by the aggregated components, and not permitting further components to aggregate to the entity – in contrast to open structures; Whitesides & Grzybowski 2002).

Desired Morphologies: are the compositions of the structures that meet a predefined set of requirements, including the arrangement of components, shape, and symmetric/asymmetric features.

The notion of encoding physical information to enable the self-assembly process is the principle contribution of this thesis. The development of this notion involves the understanding of what physical information actually means, how the description (rules) of an abstract system can be mapped using physical information to a physical system, and what is the minimal set of rules needed to enable a successful self-assembling system. The concept of physical information is further developed in *Chapter 2: Literature Review*, and further refined in *Chapter 3: Physical Information Analysis* and *Chapter 4: The Three-Level Approach to Self-Assembly Design*.

To produce evidence to support this thesis, three problems are addressed that require the creation of a self-assembly design methodology that allows for the investigation into encoding physical information. These three problems are:

1. How to specify a set of rules that can be mapped to create a physical system using physically encoded information?

2. How to automatically generate sets of rules using computer software, to address self-assembly being an algorithmically NP-complete problem?

3. How to use physical information to create more complex target structures?

The verification of the sets of rules resulting from the self-assembly design process will be conducted through physical experiments. In this work, mechanical, macroscale components (with mesoscale features) requiring external energy to permit component movement spatially in two or three spatial dimensions (2D or 3D) are used in experiments.

The corresponding environments to these components result in components either being confined to a surface (2D) or suspended in a fluid (3D), where environment vibrations transfer energy to the components.

1.3 Thesis Objectives

The objectives of the research presented, in order to test the thesis hypothesis, are the following:

1. Study existing methods to design and construct self-assembling systems.

2. Prototype the specification of rules encapsulating a self-assembling system using an idealized virtual model.

3. Analyze the first artificial self-assembling system (Penrose & Penrose 1957) through reverse engineering to identify the self-assembly rules, including their physical equivalents, used in the system.

4. Develop a self-assembly design methodology, referred to as the *three-level approach*, which inherently combines bottom-up design with bottom-up construction by using a set of self-assembly rules that can be mapped to an abstract model for analysis using software, and that can be translated to a physical system for verification.

5. Devise a set of self-assembly rules that can encode a self-assembling system, which uses passive components that can move in either two or three spatial dimensions.

6. Create two simulators to model the self-assembly rules in 2D and 3D.

7. Develop a method to fabricate components and their corresponding environments.

8. Conduct a series of experiments where a set of self-assembly rules can be specified by the user to test the three-level approach.

9. Investigate a method to which self-assembly rules can be generated bottom-up by computer software.

10. Conduct a series of experiments where a set of self-assembly rules can be generated given the attributes of a desired target structure to test the three-level approach.

11. Investigate a method that can leverage a limited set of self-assembly rules, using fixed components (that cannot differentiate), to create target structures with more complex morphologies that would not otherwise be possible.

12. Conduct a series of experiments using the method that leverages a limited set of self-assembly rules to test the three-level approach.

13. Identify the shortcomings of the conducted research, and future work.

1.4 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2: reviews the relevant literature pertaining to this thesis. An overview of self-assembly, including the types of self-assembly and classifications based on physical scale, is provided. A set of physical self-assembling systems based on the scale classification is given to show extensions to, and limitations of, systems in the literature. DNA nanotechnology and computing are described within the scope of this thesis, and are used as the foundation to the computational and complexity element to this thesis. Lastly, top-down and bottom-up self-assembly design methodologies are contrasted.

Chapter 3: provides physical information analysis in the context of self-assembly. The chapter is divided into two parts. In the first part, an abstraction of the self-assembly principles from natural and artificial systems is presented. These principles are used to design and analyze interaction rules between virtual homogeneous spherical components into target structures. In contrast, the second part analyzes the physical information in the first artificial physical self-assembling system (Penrose & Penrose 1957), where the motion of the system is restricted spatially to one dimension (1D).

Chapter 4: introduces the developed self-assembly design methodology, the three-level approach. The three levels – (1) definition of rule set, (2) virtual execution of rule set, and (3) physical realization of rule set – are described in the context of self-assembly in 2D and 3D. Different physical information encoding schemes are presented at level three to demonstrate how potential errors in the self-assembly process can be reduced or prevented.

Chapter 5: presents a programming paradigm to self-assembling systems based on physically encoded information, and uses the three-level approach directly. Eight experiments (three experiments in 2D and five experiments in 3D) are provided to test the feasibility of the physical component encoding schemes and their corresponding environments in the self-assembly of closed target structures with desired morphologies.

Chapter 6: incorporates evolutionary computing into the three-level approach to evolve self-assembly rule sets. Five experiments (two experiments in 2D and three experiments in 3D) are used to test the automated generation of component sets, and their ability (in the context of their corresponding environments) to construct closed target structures with desired morphologies.

Chapter 7: extends the original self-assembly rules presented by dividing the self-assembly process into time intervals. Four experiments (one experiment in 2D and three experiments in 3D) are provided. These experiments are used to test the self-assembly of closed structures with desired morphologies by dividing the self-assembly process into time intervals based on physically encoded information.

Chapter 8: summarizes the results of the experiments conducted, analyzes a design methodology using an example self-assembling system where the target structure has a more complex desired morphology, and offers future work extending the research presented in chapters three to seven. The design of more complex target structures is shown by how a limited set of components can be leveraged by reintroducing component information used earlier at later time intervals in the self-assembly process. Future work includes identifying the advantages of alternative component morphological information to address fault tolerances resulting during the self-assembly process, and more broad directions of self-assembly research with an exploration of error correction methods and how to scale self-assembling systems. Lastly, applications of the work presented in this thesis are discussed.

Chapter 9: concludes the thesis. Satisfaction of the thesis hypothesis is addressed by examining the experimental evidence. Lists of the scientific contributions, and publications, resulting from the research conducted as part of this thesis are provided. All of the research presented in chapters three to seven have been published or are accepted for publication. Finally, concluding remarks are given to demonstrate how the state of the art is advanced by designing self-assembling systems via physically encoded information.

Chapter 2

Literature Review

The origins of self-assembly as a research field can be traced back to organic chemistry (Whitesides & Grzybowski 2002). Since its initial formal study, the process of self-assembly has been used in a wide variety of fields and applications. The relevant self-assembly literature is critically reviewed in the context of this thesis.

Although there is no definitive definition to what self-assembly is (Bhalla 2009), the definition of self-assembly used in this thesis (*section 1.2: Thesis Hypothesis*) is the basis to the work presented, and is also specified for future comparative study and refinement. There are two main types of self-assembly, *static* and *dynamic* (Whitesides & Grzybowski 2002). Static self-assembly refers to processes that lead to structures or patterns in local or global equilibrium and do not dissipate energy (e.g. crystals). In contrast, dynamic self-assembly refers to processes that lead to structures or patterns that can only occur when the system is dissipating energy (e.g. biological systems). Research in static self-assembly is more mature, whereas research in dynamic self-assembly is in its infancy. Going from static self-assembling systems to dynamic self-assembling systems (e.g. the creation of biological cells) is even less well understood (Whitesides & Grzybowski 2002).

Further types of self-assembly are defined in (Whitesides & Grzybowski 2002), which can be considered as specializations of both static and dynamic self-assembly, and include: *templated self-assembly*, *biological self-assembly*, and *netted systems*. Templated self-assembly uses regular features in the environment to enable the self-assembly processes. Biological self-assembly is characterized by the variety and complexity of the structures, functions, and systems it produces. Netted systems consist of sensors and controllers that interact and self-assemble through data communication (e.g. robotic systems). In this work, three additional forms of self-assembly are defined, *software self-assembly*, *algorithmic self-assembly*, and *staged self-assembly* (also known as *hierarchical self-assembly*). Software self-assembly uses software components that are integrated together based on self-assembly to create specific software architectures (Li et al. 2006, 2008). Algorithmic self-assembly is where computation is embedded into the self-assembly process, typically as a process of crystalline growth (Winfree 1995; Winfree et al. 1998b). Staged self-assembly is where the self-assembly process is divided into time intervals (stages; Demaine et al. 2008). At each time interval, components can be added to, or removed from, an environment. The advantage of staging is that it encodes the construction of a target structure in the staging algorithm itself, and not exclusively in the design of components.

As presented in *section 1.2: Thesis Hypothesis*, physical, static self-assembling systems will be used in part to test the thesis hypothesis. Although the focus is on static systems, relevant physical systems are reviewed in the next section. Systems presented at the nanoscale are further discussed in the context of physical information encoding and DNA computing to underpin the theoretical aspects of self-assembly. Lastly, design methodologies that are applicable to creating self-assembling systems are contrasted.

2.1 Physical Self-Assembling Systems

Starting from the first artificial mechanical self-assembling system to state-of-the-art DNA self-assembling systems, the relevant physical self-assembling systems from the macroscale, across the mesoscale, and down to the nanoscale are critically reviewed in this section. These systems vary in terms of spatial dimension, methods to enable the self-assembly process, environment, techniques to reduce or prevent errors from occurring during the self-assembly process, and the creation of open and closed structures.

2.1.1 Macroscale Self-Assembly

At the macroscale, two broad categories of self-assembling systems are considered, mechanical systems and robotic systems. Mechanical systems use physical or chemical features in components or their environment to enable the self-assembly process. In contrast, primarily data communication is used between components (robotic units) to enable the self-assembly process of modular robotic systems. With the addition of Penrose (1958), Miyashita et al. (2009), Mao et al. (2002), Kalontarov et al. (2010), Olson et al. (2007), Tibbits (2010), Hawkes et al. (2010), Goldstein et al. (2005), and Zykov et al. (2008) the self-assembling systems reviewed here at the macroscale are considered as significant contributions to the literature (Groß & Dorigo 2008).

Mechanical Systems

Lionel S. Penrose and R. Penrose were the first to create an artificial, mechanical self-assembling system (1957). Their system consisted of two types of components, which were confined to a track environment. The track was shaken horizontally in 1D. Only in the presence of a two-component *seed complex* (consisting of both types of components) would self-assembly occur by components hooking and latching together, resulting in the self-reproduction of the seed complex. Detailed analysis of this pioneering system is provided in *section 3.2: A Self-Reproducing Analogue.* Building upon his original system, L.S. Penrose went on to extend his mechanical self-reproducing systems to 2D as layered 1D systems (Figure 2.1 I; Penrose 1958, 1959).

As a transition from 1D systems to 2D, Hosokawa et al. (1996) presented two systems using triangle shaped components that were in a flat box environment, which was rotated vertically (Figure 2.1 II). The goal of both systems was to create multiple hexagon target structures. In the first system, one permanent magnet of opposite polarity was embedded vertically in two of the three sides of each component. However, poor yield of the target

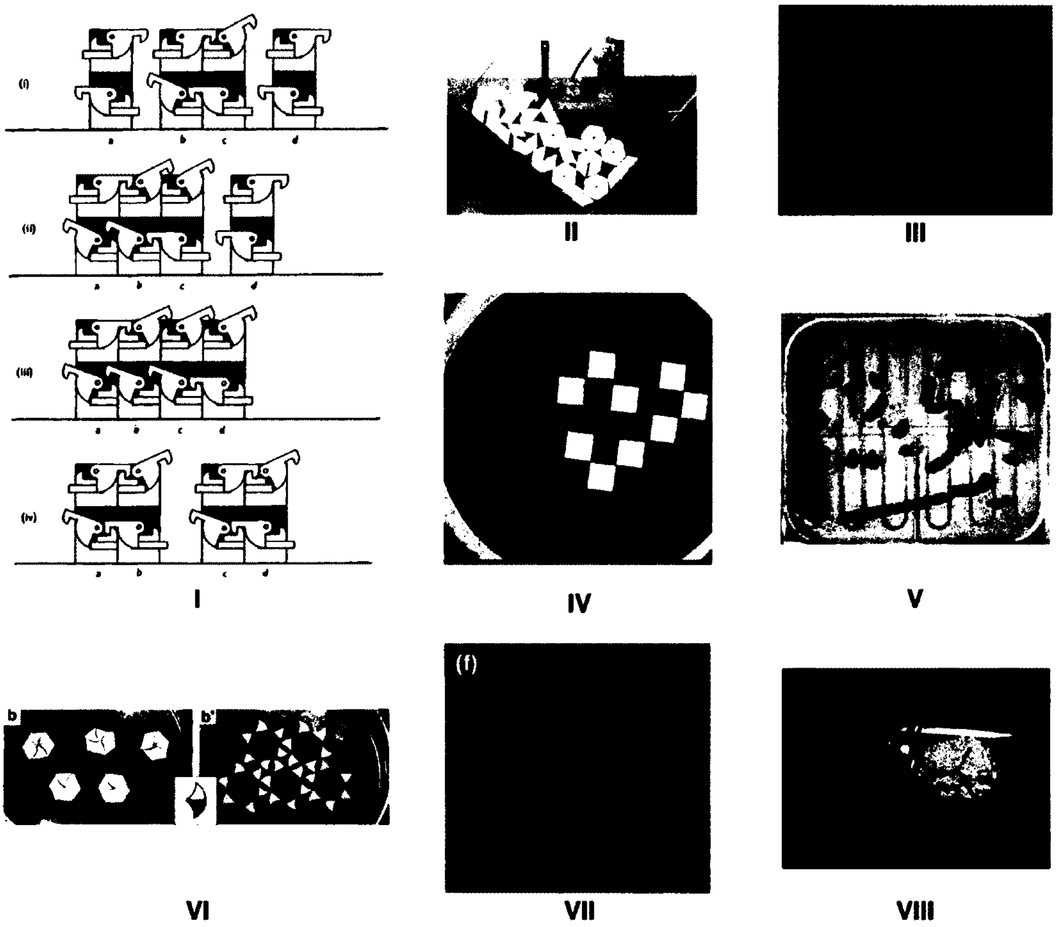


Figure 2.1: Mechanical macroscale self-assembling systems: (I) extension to the original self-reproducing system (Penrose 1958), (II) hexagon target structure (Hosokawa et al. 1996), (III) using component shape and magnetic patterns to create target structures with symmetric/asymmetric features (Bhalla & Bentley 2006), (IV) open target structures (Miyashita et al. 2009), (V) self-replication using an oscillating environment (Breivik 2001), (VI) self-reconfiguration (Reprinted (adapted) with permission from Mao et al. 2002, © 2002 American Chemical Society), (VII) cube components directed using the environment (Kalontarov et al. 2010), and (VIII) virus model (Olson et al. 2007).

structures occurred, since substructures would not necessarily self-assemble together to form target structures. To resolve this incompatible substructure problem, a second system was developed that used *seed components* and *variable components.* The number of seed components present equalled the number of desired target structures. As with components in the first system, permanent magnets of opposite polarity were embedded vertically in two edges of each seed component. However, variable components had two permanent magnets within the interior of components that would only move to the edge when a magnet of opposite polarity was present in a neighbouring component. The use of seed and variable components allowed for the creation of an assembly sequence, and prevented the occurrence of incompatible substructures. However, the drawback of this second system is that seed components were able to self-assemble, resulting in errors due to the combination of simple magnetic bonding and component shape.

As an alternative to relying on seed components for 2D self-assembly along with the prospect of creating a variety of target structures with symmetric and asymmetric polygon forms (Figure 2.1 III), the combination of permanent magnetic patterns and component shape was considered (Bhalla 2004; Bhalla & Bentley 2006; Kaewkamnerdpong et al. 2007). 2D permanent magnetic discs naturally form linear, triangular, or grid patterns. These magnetic patterns were leveraged by horizontally placing a single permanent disc magnet in the interior of components constructed from foam board. The foam was cut to create polygon key and lock shapes, similar to the hook-and-latch mechanism in L.S. Penrose and R. Penrose's original system. Closed target structures could be created by varying the size of components, and creating regions where magnets of opposite polarity would be suitably spaced as to not be able to overcome the force of friction (between components and their environment) to self-assemble. Homogenous, heterogenous, or a combination of component shapes could be used to create target structures with symmetric/asymmetric features. Components were placed on the surface of a tray. Components and their environment were hand-built. The disadvantage of this system is that the environment was shaken by hand (in 2D), limiting reproducibility.

In Miyashita et al. (2009), a single permanent disc magnet was placed in the centre of each component. Components with square, circle, and square with rounded corners were used (either a single component shape or mixed component shapes) to determine the influence of component shape on the self-assembly of open target structures (Figure 2.1 IV, page 18). Components with rounded edges (circles or squares with rounded corners) were found to be superior in experiments, since the rounded edges permitted complementary components to rotate onto one another. The environment, a tray where components floated on the surface of water, was automatically controlled. A measure, the *degree of parallelism* (DOP) was proposed to quantify the aggregation characteristic of components into a single open structure. The disadvantage of this measure is discussed in *section 2.2.5: The Complexity of Self-Assembly.*

As an alternative to using component features, the environment can also be used to enable the self-assembly process. In Breivik (2001), two component types with two different kinds of permanent magnets were used to create template-replicating structures (Figure 2.1 V, page 18). The two types of magnets vary in Curie points (the temperature above which ferromagnets become paramagnetic, and this effect is reversible). By cycling the environment temperature, components could undergo assembly and disassembly actions, creating replicas of linear seed structures. Another method using the environment to enable the self-assembly process was to change the environment chemical conditions (Figure 2.1 VI, page 18; Mao et al. 2002). For example, components with hydrophobic and hydrophilic coatings on their edges could form one type of target structure in one chemical solution, and form another target structure in another chemical solution. However, components were tethered together using string to reduce the time required to complete the self-assembly process. The environment can also be used to direct component movement. In Kalontarov et al. (2010), floating cube-based components self-assembled to a template at the bottom of their environment. Valves in the template could be opened and closed, resulting in the opening and closing of valves on the faces of components. The resulting vortices in the fluid environment was used to pull components in sequence to appropriate locations to create closed 3D target structures with symmetric/asymmetric features (Figure 2.1 VII, page 18). The advantage of using the environment is that it reduces the time required to complete the self-assembly process. However, the disadvantage of relying on the environment to enable the self-assembly process is that it is then difficult to prevent disruption leading to unwanted substructures (e.g. interrupting the self-replication process) in an automated procedure, and using global environmental conditions to direct the local interactions of many components in parallel.

3D static self-assembly has also been demonstrated using homogeneous components shaken by hand in a jar to create a single spherical structure, based on the structure of a virus (Figure 2.1 VIII, page 18; Olson et al. 2007). 3D static self-assembly has also been demonstrated by hand shaking linear strands of components that can fold into 3D structures (Tibbits 2010). Folding is directed by the arrangement of the components in their initial linear structure that can either bend left or right, implementing physical logic gates. The drawback of these two systems and Bhalla & Bentley (2006), is that hand controlled environments limit reproducibility. As well, a certain amount of *intelligent shaking* by hand might influence the results of the self-assembly process.

Robotic Systems

Robotic self-assembling systems are often distinguished by the components (robotic units) used. Passive components cannot propel themselves in their environment, whereas active components are able to propel themselves in their environment (Groß & Dorigo 2008).

Homer Jacobson used two components (referred to as heads and tails) to create a self-replicating system (1958). Components could autonomously move around a circular track, with several sidings. Initially, components were placed in a random sequence on the track. With the addition of a seed complex (consisting of one head and one tail component) the system would self-assemble replicas of the seed complex until all the components were exhausted. Templated self-assembly resulting in self-replication has also been achieved using passive components to create longer linear target structures in 2D (Figure 2.2 I; Griffith et al. 2005) and in 3D (Zykov et al. 2005). An extension addressing the limitations of replicating linear structures to various 2D shapes has been proposed in simulations using modular self-assembling robots (Mathews 2008).

Passive systems have also been created in 2D that can create various target structures (Figure 2.2 II; Bishop et al. 2005; White et al. 2004). Furthermore, these systems used components with electromagnets that could self-assemble and self-disassemble, resulting in self-recofigurable systems. The advantage of these systems is in their use of logic, via data communication between components, to enable the self-assembly process. The disadvantage of these systems is their high energy requirements (onboard components) to operate the electromagnets. 2D sheets that can fold into 3D shapes (based on origami) using electromagnets have also been developed (Figure 2.2 III; Hawkes et al. 2010), also energy intensive for folding. Consequently, these systems are not practical as they will not be scalable to systems with many components due to their energy requirements.

The use of electromagnets in passive components has also been extended to 3D. In White et al. (2005), a template at the bottom of a tank of fluid was used to fix a component to provide power through an electrical connection, as an alternative to energy onboard components (Figure 2.2 IV). The electromagnets were then used to assemble free-floating components, and disassemble components that were part of the structure. An extension to this 3D system uses valves within components and valves within the

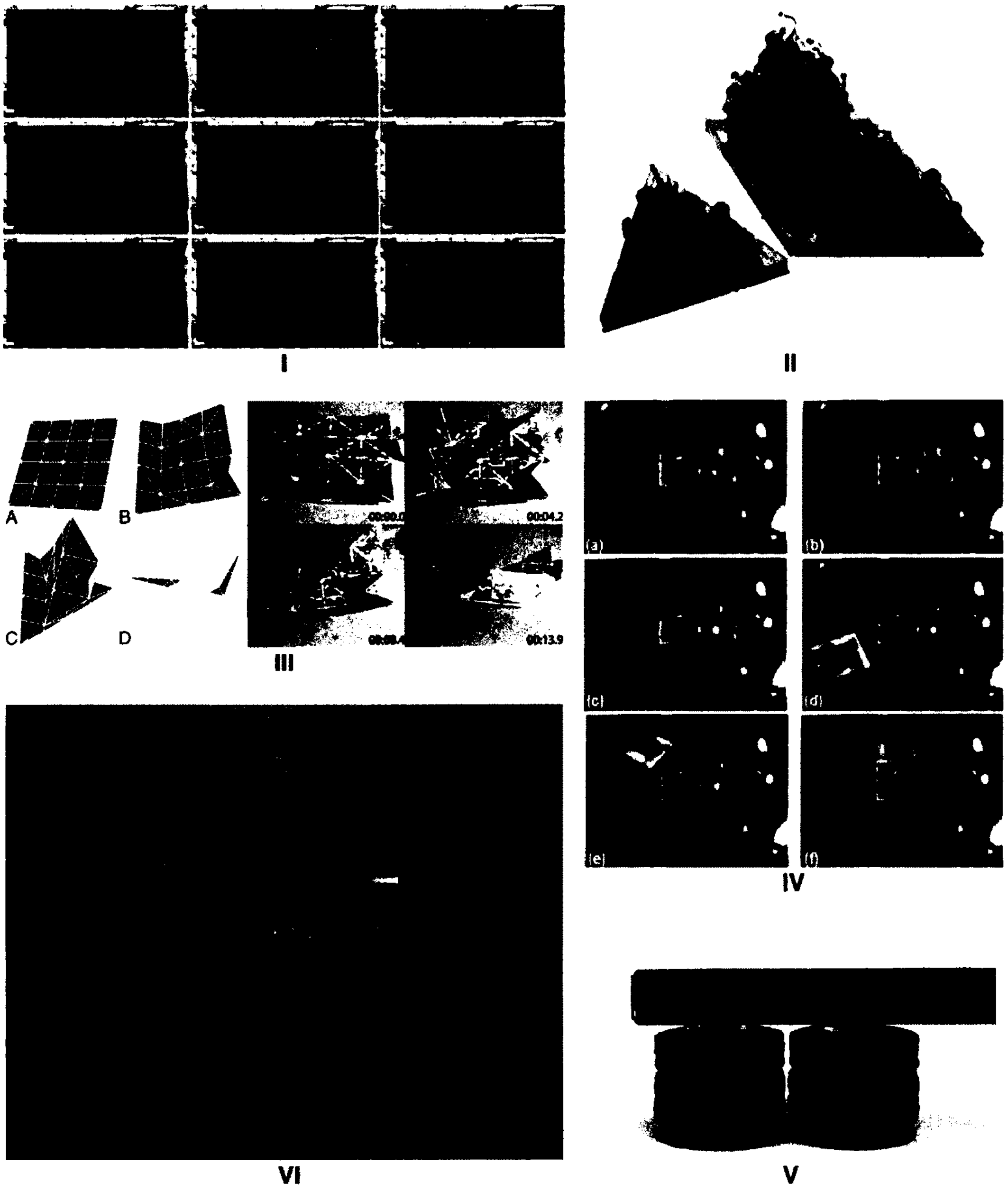


Figure 2.2: Robotic macroscale self-assembling systems: (I) 2D self-replication (Griffith et al. 2005), (II) reprogrammable components (Bishop et al. 2005, © 2005 IEEE), (III) folding sheet (Hawkes et al. 2010), (IV) 3D templated self-assembly (White et al. 2005), (V) Claytronics self-reconfigurable robot (Goldstein et al. 2005), and (VI) Swarm-bot (Mondada et al. 2003, © 2003 IEEE).

template in the environment to direct components to self-assemble to one another (White et al. 2005), with the advantage of reducing the components' energy requirements. These systems also have the disadvantage of requiring high amounts of energy to enable self-reconfiguration, and are not scalable to systems requiring many components.

Examples using active components include those which use cube-based components (varying in degrees of rotational freedom of faces) of pre-configured chains of component performed self-reconfiguration (Fukuda et al. 1991; Yim et al. 2000; Castano et al. 2000; Murata et al. 2002; Zykov et al. 2008). Cylindrical components using layers of electromagnets in pre-configured states have also been used to for self-reconfiguration (Figure 2.2 VII; Goldstein et al. 2005). Alternatively, Miyashita et al. (2010) used a *freeze-thaw connector* to freeze water between components for assembly.

However, Super Mechano Colony (SMC) (Damoto et al. 2001; Hirose 2001; Hirose et al. 2000) and Swarm-bot (Mondada et al. 2003; Dorigo et al. 2006) used active components that did not need to be pre-configured. SMC consists of *parent* and *child* components. Child components can disassemble, perform a task, and reassemble at a later time. Swarm-bot consists of homogenous cylindrical components (s-bots), and uses swarm intelligence for localized communication (Figure 2.2 VI; Bonabeau et al. 1999). The advantage of Swarm-bot over SMC was in creating static structures (Groß et al. 2006) and dynamic machines (e.g. the formation of a chain of s-bots capable of traversing a gap larger than an s-bot in the environment, and group transport of heavy objects; O'Grady et al. 2005; Nouyan et al. 2006). Using light primarily to communicate, Swarm-bot is easier to program than other robotic system. However, Swarm-bot is not scalable to 3D due to the design of the s-bots. Swarm-bot has been extended to a system using heterogeneous robots, called Swarmanoid (Mathews et al. 2010). Swarm intelligence has also been investigated as a control mechanism to enable the self-assembly of nanoscale robots (Kaewkamnerdpong 2008; Kaewkamnerdpong et al. 2007).

2.1.2 Mesoscale Self-Assembly

At the mesoscale, both millimetre and micrometre scale self-assembling systems are reviewed in 2D and 3D.

2D Systems

Capillary forces can be used to self-assemble mesoscale components (Whitesides & Boncheva 2002). In Choi et al. (2000), two type of components (triangular and circular) floating in a liquid-air interface created an open target structure where triangular components (with millimetre features) self-assembled around circular (centimetre scale) components (Figure 2.3 I). Capillary forces have also been used with hexagonal components (millimetre scale) to create an open target structure, with deliberate hexagonal holes (Figure 2.3 II; Bowden et al. 1997). Hexagonal, micrometre components have also been used to create an open, layered target structure (Clark et al. 2001). Templates have also been used to demonstrate the creation of closed target structures using hexagonal, millimetre components (Clark et al. 2002). However, using capillary forces exclusively to enable the self-assembly process is difficult to leverage for the creation of closed target structures with symmetric/asymmetric features.

Hydrophobic and hydrophilic interactions between components can also be used to enable the self-assembly process (Rothemund 2000). In this example, 2D components (cut using a laser cutter and with either hydrophobic or hydrophilic coatings on their edges) would self-assemble into periodic and aperiodic tilings (Figure 2.3 III). The goal of the aperiodic tilings were to demonstrate computation by self-assembly. The significance of aperiodic tilings is further discussed in *section 2.2.1: Physical Information Encoding in Nature* and *section 2.2.3: The abstract Tile Assembly Model.* However, the target structures (tiling of the plane using different patterns) was not achieved, since no method was used to enforce the correct sequences for tiling the plane. As a result, holes would

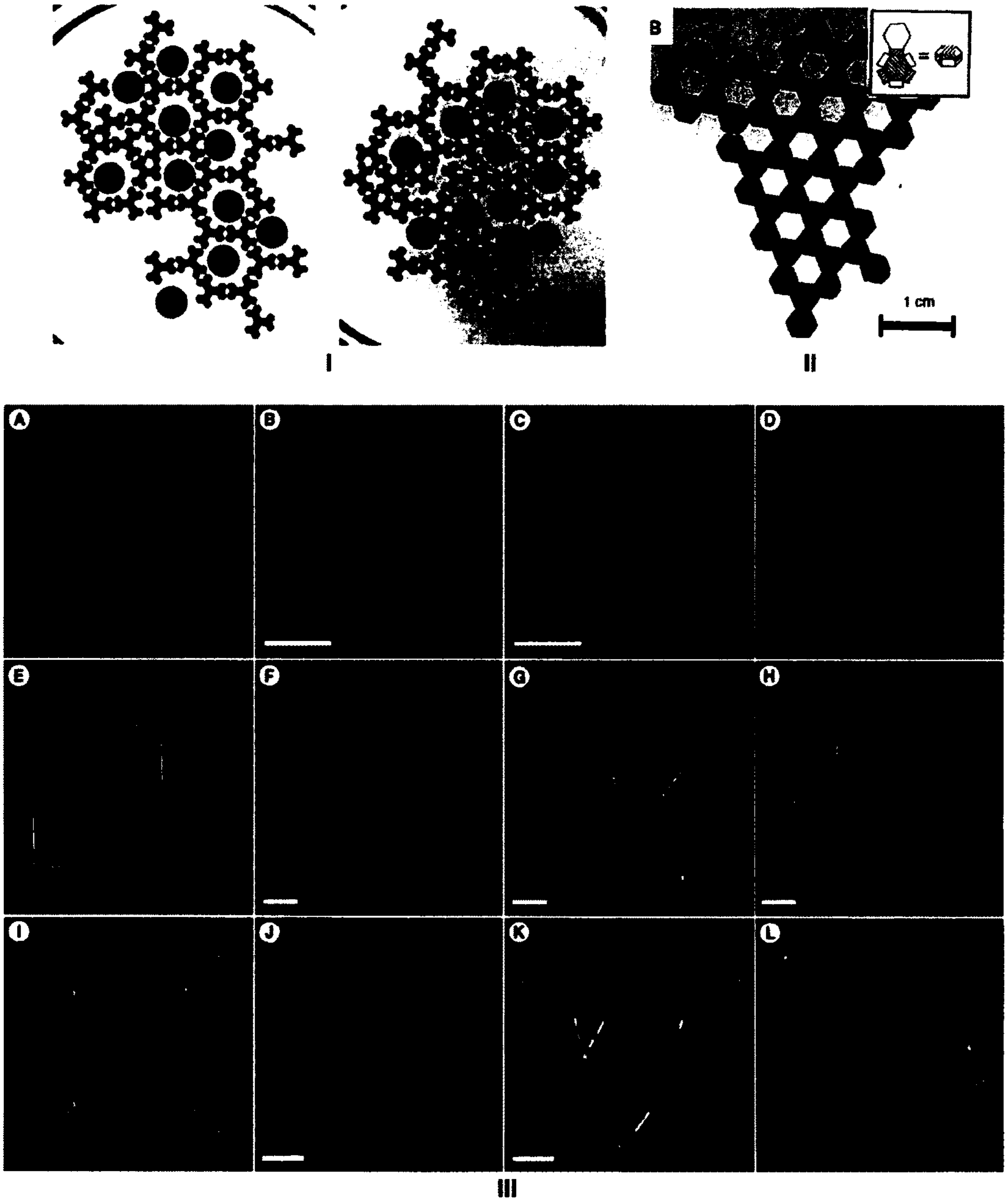


Figure 2.3: 2D mesoscale self-assembling systems: (I) triangular and circular components (Reprinted (adapted) with permission from Choi et al. 2000, © 2000 American Chemical Society), (II) open structure with deliberate hexagonal holes (Bowden et al. 1997), and (III) aperiodic tiling (Rothemund 2000).

form in the structure or incompatible substructures would form.

The use of environmental conditions has also been used to enable self-assembly using millimetre scale and micrometre scale components. As an example of dynamic self-assembly using mechanical components, permanent magnetic discs floating either in a liquid-air interface (Grzybowski et al. 2000), floating in a layered formation (Grzybowski & Whitesides 2002b), or using chiral spinners (Grzybowski & Whitesides 2002a) would self-assemble into different target structures due to an electromagnet in the environment. Various self-assembled formations would occur based on the rotation sequence of the electromagnet. The self-assembled structures would no longer exist when the electromagnet was turned off (when the system was no longer dissipating energy). Another example of dynamic self-assembly is in Diller et al. (2011), where an external magnetic field was oscillated to enable the self-assembly and locomotion of micrometre scale robots on a 2D surface.

Changes in environmental conditions have also been used to create static structures using micrometre scale components (Krishnan et al. 2008; Tolley et al. 2008). By controlling the flow in a microfluidic chamber, components could be directed globally and use local interactions to self-assemble. This technique of controlling flow, along with templates in the environment, has successfully created both symmetric and asymmetric target structures. An extension to this system was presented in (Krishnan et al. 2009), where components could be selectively assembled to, or disassembled from, a larger structure by direct thermal modulation of the local viscosity field surrounding components. The advantage of these systems, including Diller et al. (2011), is that they do not rely on pre-programmed assembly protocols between the components. The challenge with these techniques is when scaling to systems with large numbers of components, where again it is difficult to use global conditions to direct local interactions.

3D Systems

Terfort et al. (1997) were the first to demonstrate 3D artificial self-assembly using millimetre scale components (Figure 2.4 I). Three systems used homogeneous components to create symmetric structures, and one system used two different components (one with a key shape and one with a lock shape) to create a cylinder. A variety of open target structures using helical, millimetre scale components have also been created (Figure 2.4 II; Boncheva et al. 2003a). In contrast to Boncheva et al. (2003a), Terfort et al. (1997) demonstrated the advantage of using hydrophobic and hydrophilic properties in combination with two types of component shapes in one system to enable the self-assembly process using heterogeneous components for the self-assembly of a closed target structure.

Staging has also been used to enable the self-assembly of millimetre scale spherical components into various open target structures (Figure 2.4 III; Wu et al. 2002). Two time intervals were used. In the first time interval, spherical components were packed into templates (tubes with triangular, hexagonal, rectangular, and circular shapes). The second time interval used the substructures created in the first time interval to create the open target structure (e.g. using linear substructures to create a cubic lattice open target structure). Furthermore, templates in the second time interval could also be used to create open target structures (e.g. hexagonal shaped tube to create a hexagonal lattice open target structure). In this example, only templates at each stage were used to enable the self-assembly process. No examples of using component properties at each stage to enable the self-assembly process were provided.

Garcias et al. (2002) created a system where the faces of self-assembled micrometre cubes had patterned faces (four metallic squares near each corner of a face, Figure 2.4 IV). The resulting cubes were constructed using a folding technique, where the faces of a cube were initially attached in a 2D arrangement. However, the patterning on the faces of components did not enable the self-assembly of cubes into larger structures. As

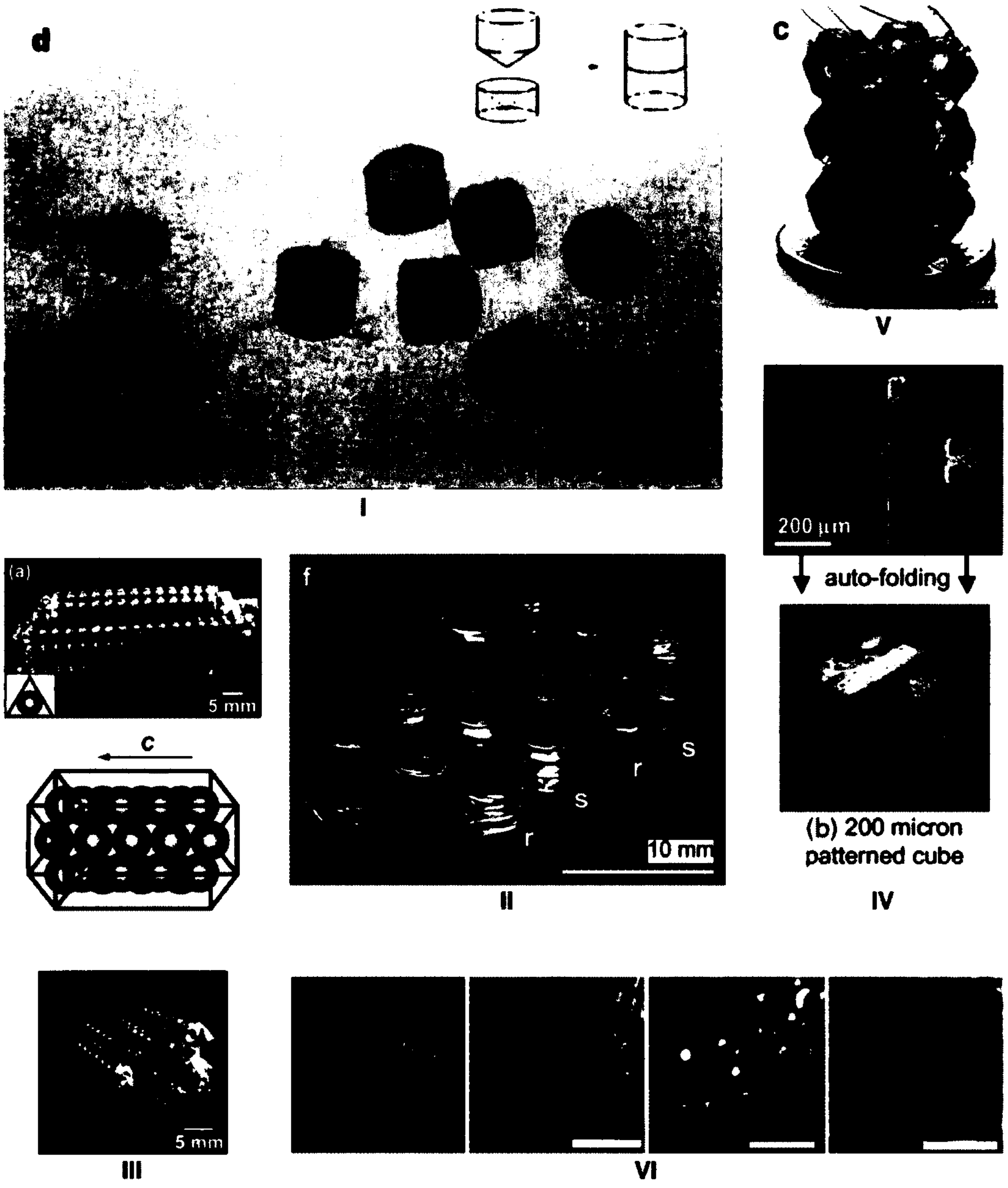


Figure 2.4: 3D mesoscale self-assembling systems: (I) cylinder using key-lock shape (Terfort et al. 1997), (II) open chiral structure (Reprinted (adapted) with permission Boncheva et al. 2003a, © 2003 American Chemical Society), (III) staged self-assembly using spherical components (Reprinted (adapted) with permission Wu et al. 2002, © 2002 American Chemical Society), (IV) cube with patterned faces (Garcias et al. 2002), (V) 3D circuit (Garcias et al. 2000), and (VI) casquet made using micro-masonry (Fernandez & Khademhosseini 2010).

a precursor to this work, polyhedral components were self-assembled in an open lattice structure creating a 3D circuit (Figure 2.4 V; Garcias et al. 2000).

A technique referred to as *micro-masonry* created 3D target structures using shape-controlled microgels (Fernandez & Khademhosseini 2010). Microgels self-assemble using capillary forces, on the surface of a template. Multi-layered structures can be created using microgels using components with key and lock shapes. Micro-masonary has been used to create a tube, solid ball, and casquet target structures (Figure 2.4 VI). The advantage of this system is that it uses a combination of capillary forces and a template to create closed target structures, with symmetric features.

2.1.3 Nanoscale Self-Assembly

At the nanoscale, DNA is considered as one of the most promising materials for the creation of nanotechnology due to its inherent self-assembly properties (Seeman 2004, 2007). DNA nanotechnology was invented by Nadrian Seeman (Pelesko 2007), who realized that 3D lattices could be used to direct molecules simplifying their crystallographic study. The first 3D nanoscale objects created using self-assembly were a cube and a truncated octahedron made out of DNA (Figure 2.5 I; Chen & Seeman 1991; Zhang & Seeman 1994). However, these objects were not rigid enough to create 3D lattices (Pelesko 2007).

DNA tiles were developed to address this rigidity problem, while still being able to create lattice structures (Winfree et al. 1998a). These tiles use interwoven double DNA strands to create the square body of a tile, with single DNA strands extending from the edges of a tile (Figure 2.5 II). A tile type is defined by the single strands (motifs) extending from the North, West, South, and East edges of the 2D DNA tiles. At the time of writing, the realization of 3D DNA tiles has not been achieved.

Paul Rothemund pioneered a technique to create a wide variety of 2D self-assembled DNA structures, referred to as *DNA origami* (2005; 2006). This technique uses smaller

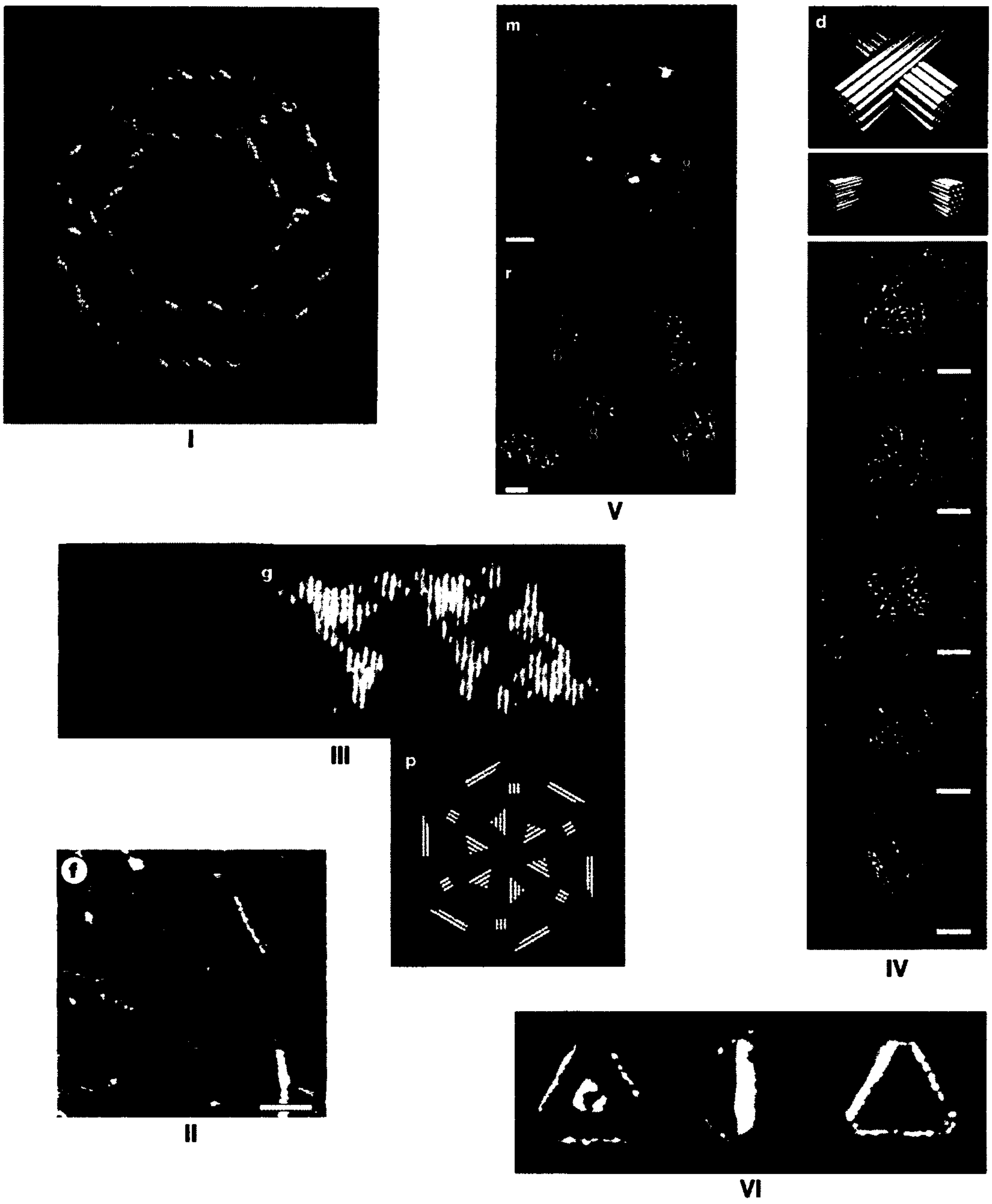


Figure 2.5: DNA nanoscale self-assembling systems: (I) truncated-octahedron (Reprinted (adapted) with permission Zhang & Seeman 1994, © 1994 American Chemical Society), (II) DNA Tiles (Winfree et al. 1998a), (III) 2D DNA origami (Rothemund 2006), (IV) 3D DNA origami (Douglas et al. 2009), (V) 2D DNA origami using shapes (Woo & Rothemund 2011), and (VI) staged tetrahedron (He et al. 2008).

single strands of DNA to act as scaffolding, which self-assemble to a longer single strand of DNA folding it into a target structure (Figure 2.5 III). The use of folding in DNA origami has been extended to 3D (Figure 2.5 IV; Douglas et al. 2009). However, only one example was shown in Douglas et al. (2009) where components created using 3D DNA origami would then self-assemble into a 3D target structure. In this example, identical components were used to self-assemble multiples of a symmetrical target structure, a tetrahedron.

Methods to scale DNA nanotechnology to use larger numbers of components to create more complex target structures include using error correction techniques (Kari & Mahalingam 2010; Meng & Kashyap 2009) and staged self-assembly (Demaine et al. 2008). One of the drawbacks of DNA tiles is the challenge in designing the extending single strands. For large numbers of tiles, it is difficult to create a variety of single strands that do not have complementary regions (between separate strands or within an individual strand). As a result, long single strands are required. Long single strands of DNA pose a problem as they are no longer rigid in comparison to short single strands, and cause defects to occur during the self-assembly process. One solution to this problem has been proposed using DNA origami to design shapes, acting as keys and locks, within components. Single DNA strands within these shapes are used to assemble complementary shapes between components (Figure 2.5 V). The combination of these shapes and short single DNA strands is used to reduce errors from occurring during the self-assembly process (Woo & Rothemund 2011). This techniques has not been extended to 3D, nor has a similar technique in the creation of 2D DNA tiles using shapes (Turberfield 2011). Likewise, He et al. (2008) used three-point star motif tiles in a two-staged self-assembly process to reduce errors and create more complex target structures (Figure 2.5 VI). By controlling the motif lengths and the concentration of tiles, three-point star tiles were able to self-assemble into tetrahedrons, dodecahedrons, and buckyballs. However, the combination of these two techniques using component morphology and staged self-assembly has not been used.

DNA is also being used to devise nanoscale robots, nucleic acid robots (*Nubots*). DNA hybridization can be used as a chemical energy source to drive these molecular machines (Turberfield et al. 2003). Example machines include a *DNA motor* (Yurke et al. 2000), a *DNA glider* (Zhang & Seelig 2011), and a *DNA walker* (Omabegho et al. 2009).

2.2 Physical Information Encoding and DNA Computing

Advancements in DNA nanotechnology, particularly the invention of DNA tiles, has allowed for investigations into new forms of embodied computation. The computational perspective of these advancements, including physical information encoding in nature, DNA computing, a DNA tiling model and its extension, as well as the complexity of self-assembly are all considered in investigating, what is physical information?

2.2.1 Physical Information Encoding in Nature

One of the most prolific insights was that of Erwin Schrödinger and his proposed aperiodic crystal for genetic information encoding, described in his popular science classic, *What is Life?* (1944, reprinted 2003). Schrödinger conjectured that in contrast to a periodic crystal, only an aperiodic crystal would be able to create a microcode for an organism. Schrödinger's aperiodic crystal predates the discovery of the aperiodic structure of DNA (Symonds 1986, 1987), later discovered by James Watson and Francis Crick (1953).

Crick acknowledges Schrödinger for inspiring him to study the structure of DNA (Murphy & O'Neill 1997). The discovery of the structure of DNA lead to the central dogma of molecular biology (Crick 1970). The central dogma describes the transfer of genetic information encoded in DNA, through transcription to RNA, and translation to Proteins. Proteins, the resulting self-assembling shapes, are the primary building blocks of living organisms. It is this mechanism of genetic information transfer that enables the process of biological development, which uses explicit stages in its provision of a solution to the construction of a multicellular organism (Wolpert 1998). The explicit stages in biological development are often irreversible, and cannot be repeated at later stages, such as invagination, gastrulation, and the formation of a body plan. This higher-order process of staged development in nature allows for the creation of more complex phenotypes, which would not otherwise be possible (Wolpert 1998). Driving the mechanisms behind both the process of genetic information transfer and the process of staged biological development is molecular recognition. Over a century ago, Emil Fisher proposed the key-lock principle to explain the mechanism of enzyme action (Ball 1994). The idea was that an enzyme has a bonding site that fits the shape of its substrate. Shape and geometry are central to molecular interactions, within the context of an appropriate environment. Fisher's principle is still valid today, and the analogy is applicable to explaining molecular recognition.

However, it was Schrödinger's insight into the requirement of an aperiodic crystal containing a microcode that could generate an organism that forms the basis for Stuart Kauffman's question[1], *what is information?*. In Kauffman's opinion, traditional definitions to what information is, such as Shannon information (Shannon 1948) and Kolmogorov information (Kolmogorov 1965), are insufficient to describe the complexity of the biosphere as they have no direct linking of information to matter and energy in these definitions. From Kauffman's perspective, the notion of *generating* is the set of specific processes aspect of information that is needed in the definition of what information *is*, as part of the explanation to what lead to the complexity of the natural world. The inclusion of generation of physical processes differentiates this notion of physical information from that used in physics (Landauer 1996, 1999; Friedman 1998, 2001), and will be fur-

[1] S. Kauffman (2010). *What is information?* NPR 13.7 Cosmos and Culture. URL: http://www.npr.org/blogs/13.7/2010/06/04/127473541/what-is-information

ther discussed in *section 2.2.5: The Complexity of Self-Assembly* from the perspective of DNA computing.

2.2.2 DNA Computing

In addition to understanding how the aperiodic structure of DNA serves as a genetic information medium in biological systems, research is being conducted on manipulating DNA for information encoding and computation. Leonard Adleman first showed that computing using self-assembling molecules, DNA, was possible (1994). An algorithm was devised to solve a seven node directed Hamiltonian path problem (Garey & Johnson 1979). Single strands of DNA were created to encode partial candidate solutions to the problem (representing the edges and vertices of the directed graph). These partial sample solutions were then allowed to interact and self-assemble based on Watson-Crick complementarity. The linear double-stranded self-assembled structures representing potential solutions were then filtered using biochemistry and molecular biology techniques to identify the true self-assembled solution.

With the success of Adleman's experiment, research has focused on scaling DNA computing to solve larger problems (Păun et al. 1998). This requires a shift from the original brute-force approach. Developing new DNA structures and algorithmic techniques has shown tremendous promise in furthering this method of embodied computation. The role of shape and structure in physical information encoding and computation plays a vital role in embodied computing, as will be discussed in the progress from the origins of DNA computing.

2.2.3 The abstract Tile Assembly Model

Wang tiles, created by Hao Wang (1961; 1965), are a mathematical construction where each unit square tile has coloured edges. Wang tiles can be arranged (translated, but

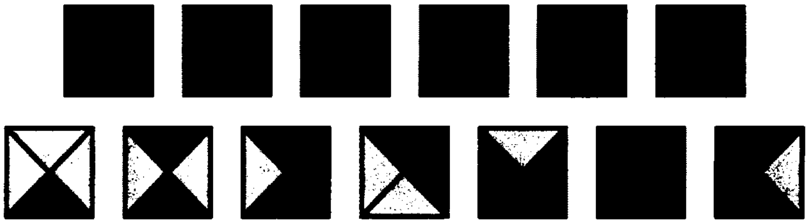


Figure 2.6: A set of 13 Wang tiles that only result in an aperiodic tiling of the plane.

not rotated or reflected) on a rectangular plane only when adjacent tiles have the same colour. Given a set of Wang tiles, the main question is proving if that set (allowing for infinite copies of tiles in the set) can tile the plane or not. Wang presented an algorithm that could take any finite set of Wang tiles and decide if that set could tile the plane or not (1961). In his proof, Wang assumed that any set that could tile the plane would result in a periodic tiling. However, Robert Berger proved that Wang's conjecture was false, by creating a set of Wang tiles that would result only in an aperiodic tiling (1966). This form of aperiodic tiling is similar to a Penrose tiling (Penrose 1974) and Quasicrystals (Shechtman et al. 1984; Senechal 2006). Berger's original set contained 20,426 Wang tiles. Later, Karel Culik (1996) created a set of Wang tiles of size 13 that would result only in an aperiodic tiling (Figure 2.6). Any Turing machine (Turing 1936) can be translated into a set of Wang tiles, such that the set of Wang tiles will only tile the plane if and only if the Turing machine will never halt. The halting problem (Sipser 1997; Turing 1936) is undecidable, just as Wang's tiling problem is uncomputable. Wang tiles have been proven to be Turing universal (Sipser 1997).

The physical realization of Wang tiles was achieved through the creation of DNA tiles, where single DNA strands extending from the edges of the square tiles represent colours. Erik Winfree created the abstract Tile Assembly Model (aTAM; 1998b), originally named the Tile Assembly Model (TAM; Rothemund & Winfree 2000), as a mathematical model of pseudo-crystalline growth.

In the aTAM, a *tile type* is defined by the *bonding domains* (e.g. colours) on the North, West, South, and East edges of a tile. A finite set of tile types is specified (which are in infinite supply in the model). Tiles can only bond together if the interactions between bonding domains are of sufficient strength (provided by a *strength function*), as determined by the *temperature* parameter. The sum of the bonding strengths of the edges of a tile must meet or exceed the temperature parameter. As a result, the temperature parameter dictates *co-operative bonding.* There are four constraints with the aTAM:

1. At least one seed tile must be specified to start the self-assembly process.

2. Tiles cannot be rotated or reflected (realized through co-operative bonding constraints).

3. There cannot be more than one tile type that can be used at a particular assembly location in the growing structure (although the same bonding domain is permitted on more than one tile type).

4. All tiles are present in the same environment, referred to as a *one-pot-mixture.*

These four constraints, along with the set of tiles, bonding domains, strength function, and temperature parameter define the aTAM. The seed tile is first selected and placed on the square lattice environment. Tiles are then selected one at a time, and placed on the grid if the bonding strength constraints are satisfied. The output is a given shape of fixed size, if the model can uniquely construct it.

There are alternative models of self-assembly (Krasnogor et al. 2008), which are based on: Artificial Chemistry (Dittrich et al. 2001), Dissipative Particle Dynamics (Groot & Warren 1997), dynamic Bonding Dissipative Particle Dynamics (Buchanan et al. 2008), P-Systems (Bernardini & Gheorgh 2004), and the Sticky Graph model (Boo 2004). However, these models in contrast to the aTAM do not link self-assembly to computation.

2.2.4 Extensions to the abstract Tile Assembly Model

There are several extensions to the aTAM (Aggarwal et al. 2005; Chandran et al. 2009; Demaine et al. 2008; Doty 2009; Kao & Schweller 2008; Patitz et al. 2011; Winfree 1998b), offering an alternative *programming* approach within tile-based systems due to their additional model features. These extensions and the aTAM can be classified into the following four programming approaches (Doty 2009):

1. Tile Hard-coding Programming (THcP)
2. Staged Programming (SP)
3. Tile Concentration Programming (TCP)
4. Temperature Programming (TP)

THcP relies on hard-coded information in the tiles to direct the self-assembly process. The aTAM falls under this category. SP allows for tiles to be added dynamically in sequence, or intermediate structures can be stored separately and added to the main mixture at later time steps. These staged additions of individual tiles or intermediate structures enables the self-assembly process in SP. In TCP, tile types are specified in terms of concentration values. The probability of tile interactions is used to enable the self-assembly process in TCP. TP uses changes or fluctuations in the environment temperature to enable the self-assembly process. Each of these extensions has been useful in investigating the impact of different aspects to the self-assembly process from an algorithmic perspective. However, none of these extensions consider the potential impact of fundamental properties, such as component rotation, on the self-assembly process (further discussed in *section 2.2.5: The Complexity of Self-Assembly*).

Lego World was created by Kauffman, which served as a model to study complexity and emergence abstractly (2000). The model generates the possible outcomes (structures) from assembling *Lego bricks.* The bricks (e.g. rectangular blocks of size 1×1, 1×2, 2×3, and 3×4) represent the *primitive set.* The *primitive operations* permitted in Lego World include both construction and deconstruction, as adding two primitive parts together or adding a primitive part to a growing assemblage (object structure), and removing a primitive part off another primitive part or assemblage. The inclusion of deconstruction steps is the primary difference between Lego World and the aTAM and its extensions.

The set specifying the quantities of the unassembled primitive bricks in the primitive set is referred to as the *founder set* in Lego World. To determine the types of structures that are possible from a founder set in Lego World, one can consider the number of steps required to reach a structure. At each step, a single primitive operation can be performed. For example, rank one includes all the possible structures that can be built from one construction step, and all the possible structures from two construction/deconstruction steps would be part of rank two. The number of ranks may continue to infinity, given an infinite founder set. Kauffman refers to the founder set and the transformations on those bricks into assemblages as a *technology graph.* Kauffman notes that a technology graph is similar to a chemical reaction bipartite graph, where the founder set is a set of organic molecules, the resulting molecules are the objects, and the transformations are the reaction hyperedges linking substrates and products among the objects (Kauffman 2000). The graph is bipartite with two types of entities: nodes (representing objects) and hyperedges (representing transformations). In this form, a technology graph can also be considered as an abstract model for self-assembly, and its relationship to chemical reactions as a parallel to the origins of self-assembly and its roots in organic chemistry (Whitesides & Grzybowski 2002).

2.2.5 The Complexity of Self-Assembly

The aTAM and its extensions have been used to investigate the complexity of self-assembly, asking primarily four questions:

1. What is the computational power of self-assembly?
2. How complex is the problem of having a set of components self-assemble into a target structure?
3. Can self-assembly be used to perform computation?
4. How does one measure the complexity of a self-assembled target structure?

Turing Universality

By basing the aTAM on Wang tiles (*section 2.2.3: The abstract Tile Assembly Model*), Winfree was able to prove that the aTAM is Turing universal at temperature two in 2D (Winfree 1995). Winfree's proof was done by showing a one-to-one-mapping between the aTAM and Wang tiles. Winfree then used the aTAM to design a set of physical DNA tiles that are Turing universal.

However, an extension to the aTAM using *restricted glue* strengths (i.e. bond strengths), referred to as the restricted glue Tile Assembly Model (rgTAM), is Turing Universal at temperature one in 2D (Patitz et al. 2011). This was achieved by:

- having an absolute value of one for every glue strength,
- using a single *negative* glue type (i.e. a repulsive interaction between components), and
- excluding conflicting glue types from interacting with one another (i.e. the glue interaction matrix is diagonal).

The term 'restricted' in the rgTAM is used to distinguish it from other extensions to the aTAM that use negative glues, as those extensions allow for non-diagonal glue functions (e.g. multiple glue types over a large range of glue strengths). The rgTAM is restricted to glue strengths of either -1, 0, or 1. As stated by the authors, the goal of the rgTAM is "to study the "simplest" model of algorithmic self-assembly that retains the computational and expressiveness of temperature 2 self-assembly" (Patitz et al. 2011). The physical realization of temperature two tile-based systems has proven to be difficult (e.g. partial attachments between components), and therefore "the characterization of self-assembly at temperature 1 is of the utmost importance" (Patitz et al. 2011). The physical creation of DNA tiles with repulsive interaction (negative glues) can be achieved by attaching magnetic particles to DNA (Kinsella & Ivanisevic 2005; Rickwood & Lund 1998).

The rgTAM was proven to be Turing universal at temperature one in 2D by first defining a *zig-zag tile assembly system* (Cook et al. 2011). Zig-zag systems are those that grow horizontally one row at a time, by alternating in either left-to-right or right-to-left growth in only one direction (e.g. north and not south). Zig-zag systems are Turing universal (Cook et al. 2011). It was then shown that there is a restricted tile assembly system in rgTAM for every zig-zag tile assembly system. The combination of positive and negative glues are used to direct the self-assembly sequence of the target structures.

The aTAM has also been extended to 3D using cube-based tiles (Zhang et al. 2011), equivalent to 3D Wang tiles (Culik & Kari 1996). In addition to the rgTAM, the 3D aTAM has been proven to be Turing universal at temperature one (Cook et al. 2011). The proof was conducted by showing how 3D systems are capable of simulating large classes of 2D systems, including 2D zig-zag systems.

NP-Completeness

The aTAM has also been used to study the complexity of the problem of self-assembly itself, using the Minimum Tile Set Problem (MTSP; Adleman et al. 2002). The goal of the MTSP is to find the lowest number of tile types that can uniquely self-assemble into a target structure. The MTSP is an NP-complete problem for general target structures. A decision problem C is NP-complete if C is in the class NP (condition one) and every problem in the class NP is reducible to C in polynomial time (condition two; Sipser 1997). Condition one can be met by proving that C can be verified in polynomial time. To meet condition one, Adleman et al. (2002) created an algorithm, *Unique-Shape*, that decides whether a tile system uniquely produces the shape of a target structure, which they proved can be verified in polynomial time. Condition two can be met by proving that an already known NP-complete problem can be reduced to C (i.e. proving that C is NP-hard). The decision problem for determining if the variables in a Boolean formula can be assigned in such a way to have the formula evaluate to true is referred to as *satisfiability* (SAT; Sipser 1997). SAT was the first decision problem known to be in the class NP-complete (Cook 1971). One of several special cases of SAT is where the formulae are in *conjunctive normal form* (CNF; Sipser 1997). The 3CNF-SAT problem (determining if each clause in a formula is limited to at least three literals) is known to be in the class NP-complete (Karp 1972; Sipser 1997). To meet condition two, Adleman et al. (2002) reduced the 3CNF-SAT problem to the MTSP by encoding a 3CNF-formula into shapes using two structures: (1) a tree, for which it is possible to compute the minimal tile set in polynomial time, and (2) a tree substructure, for which can only be satisfied if and only if the tree substructure can be assembled using distinct tile types. By proving that the MTSP is in the class NP and that the MTSP is in the class NP-hard (by proving that 3CNF-SAT can be reduced to the MTSP), Adleman et al. (2002) proved that the MTSP is in the class NP-complete for general target structures.

In addition to the MTSP, Adleman et al. (2002) also devised the Tile Concentration Problem (TCP). The goal of the TCP is to find the relative concentrations of tile types that self-assemble into a target structure using the fewest assembly steps. The algorithmic complexity of the TCP has only been calculated for specific classes of target structures. The aTAM and variations of tile-based self-assembly have also been used to investigate self-assembly complexity classes (Jonoska & McColm 2009) and using self-assembly to solve NP-complete problems (Brun 2008a,b).

Algorithmic Self-Assembly

The aTAM and its extensions have also been used to analyze the algorithmic complexity of self-assembly. The aTAM has lead to new algorithmic proposals for using DNA to perform mathematical operations, including binary counter, multiplication, and cyclic convolution product. Several binary counter algorithms have been proposed using DNA tiles that self-assemble into 2D structures (Barish et al. 2005; de Espanés & Goel 2007; Goel et al. 2004; Krasnogor et al. 2008). There has been progress on constructing a physical example of a binary counter, but with a few counting errors due to assembly errors (Barish et al. 2005). A theoretical algorithm to perform multiplication also uses a 2D structure (Pelletier & Weimerskrich 2002). The theoretical algorithm for the cyclic convolution product requires new 3D DNA tiles allowing for the self-assembly of a 3D structure (Pelletier & Weimerskrich 2002). However, the physical creation of 3D DNA tiles has not been achieved at the time of writing.

The aTAM has also been used to study the algorithmic complexity of constructing target structures. One target structure is the *Sierpinski Triangle* (Stewart 1995), which is a fractal (Mandelbrot 1977). The significance of this target structure is that it is self-similar, and the pattern can be reproduced at any magnification or reduction. As a result of using a Sierpinski Triangle as a target structure, it was physically demonstrated that small sets of components could be used to self-assemble into large (theoretically infinite) target structures (Rothemund et al. 2004).

The most common target structures (at the time of writing) used for comparative study are lines and squares (Adleman et al. 2001; Rothemund & Winfree 2000; Winfree 1998a). Lines serve as a basic target structure, and which can also be used as substructures (e.g. zig-zag systems). Squares pose a challenge, as a method of coordination is required during the self-assembly process to prevent the formation of holes.

Complexity Measures

Complexity in self-assembling systems is primarily measured in terms of Kolmogorov information (Krasnogor et al. 2008; Pelesko 2007; Rothemund & Winfree 2000). Given an input string of symbols, Kolmogorov information is the measurement of the smallest computer program that can output the input string. In this context, Kolmogorov information represents the number of tile types required to build a structure, such as a line or a square. In the case of the aTAM, the complexity of creating a line of size n is $\mathcal{O}(n)$, where $n \in \mathbb{N}$, since no rotations or reflections are permitted (Figure 2.7). If rotation is permitted then the complexity is reduced to either $\mathcal{O}(n/2)$ or $\mathcal{O}(\lceil n/2 \rceil)$ for lines consisting of either an even or odd number of tiles respectively (Figure 2.7). Although a constant factor reduction is not considered a significant improvement from an algorithmic complexity perspective, as others have shown greater complexity improvements for the creation of lines with extensions to the aTAM that continue to not permit rotation and reflection, e.g. using tile concentrations (Chandran et al. 2009). However, this simple example is used to illustrate the shortcomings of the aTAM and its extensions, and the use of Kolmogorov information to measure complexity in self-assembling systems.

When considering the construction of physical self-assembling systems, permitting rotation allows one to take advantage of any symmetry in the target structure. Leveraging

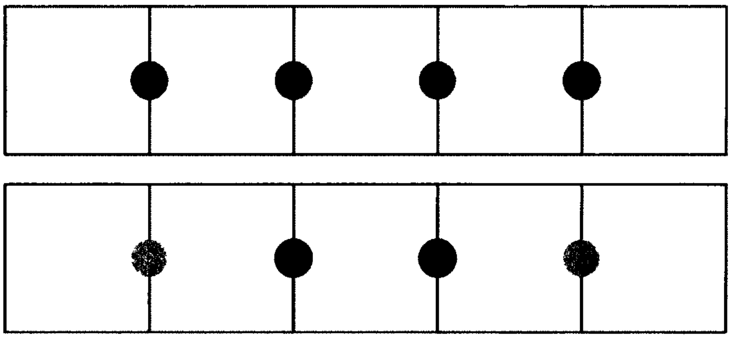


Figure 2.7: An example of the benefit of reducing the amount of information required when rotation is permitted (bottom) compared to when rotation is not permitted (top) between components with complementary information (represented using colours, similar to Wang tiles where tiles with the same coloured edges can be assembled).

symmetry in a system allows for the following benefits:

- reduce the number of tile types required,
- reduce the time it takes for a self-assembled structure to emerge by increasing the interchangeability of components in the self-assembly process, and
- reduce the number of unique bonding domains.

These advantages of leveraging symmetry, enabled by rotations, are not encapsulated using Kolmogorov information. The aTAM is useful for mathematically analyzing self-assembling systems (and permitting rotations does make the analysis more difficult). However, the disembodiment of Kolmogorov information creates a great disconnect between these tile-based models and the desired attributes in a physical system.

Likewise, Shannon information is disconnected from matter and energy. Shannon information is the measurement of transmitting a string of symbols across a channel from a sender to a receiver, e.g. the number of bits required to transmit a binary string. As such, Shannon information tells one how much information is present (or required for transmission), but does not say what that information is. The DOP (*section 2.1.1: Mechanical*

Systems), based on Shannon information, has been applied to measure the parallelism in components being able to self-assemble into open target structures (Miyashita et al. 2009). Although parallelism is an important characteristic to self-assembling systems, it is not exclusive. Shannon information and Kolmogorov information do not fully measure self-assembling systems.

2.3 Designing Self-Assembling Systems

As shown with the examples in *section 2.1: Physical Self-Assembling Systems* and as discussed *section 2.2.5: The Complexity of Self-Assembly*, creating self-assembling systems pose many difficult design challenges. In this section, top-down versus bottom-up self-assembly design is contrasted with respect to self-assembly being an algorithmically NP-complete problem.

2.3.1 Top-Down Self-Assembly Design

As with the physical self-assembly examples presented in *section 2.1: Physical Self-Assembling Systems* and further examples based on the relevant literature (Groß & Dorigo 2008), primarily top-down self-assembly design methodologies have been used to create physical self-assembling systems. Top-down self-assembly design uses the information from a target structure's global morphology to create a set of components. Algorithms evaluating the geometry of a global morphology have been used in modular robotics (Jones & Matari 2003; Klavins et al. 2006; Nagpal 2006). DNA origami has also relied on using the global morphology of target structures to act as templates, for example to dictate the mechanism in which components can recreate 2D images (Rothemund 2006, 2005).

An extension to this dissection method was presented in Tolley & Lipson (2010), for stochastic assembly planning of cube-based components in a fluid environment. A template in the environment is used to direct fluid, by opening and closing valves within components to enable the self-assembly process. A cube-based representation of the target structure's geometry is first created, by disassembling it virtually one cube-based component at a time. The reverse order of this process defines one assembly sequence that is guaranteed to result in the target structure without error. Since it is difficult to predict when and where components will be available for self-assembly, the time required to complete the self-assembly process can be decreased by planning alternative assembly sequences (opening and closing valves) depending on which options occur first. As a result, this top-down design self-assembly design methodology is capable of on-line planning.

An alternative top-down self-assembly design methodology based on the aTAM uses a spanning tree (Solveichik & Winfree 2007). Informally, a spanning tree T of an undirected graph UG is a selection of edges from G that form a tree spanning every vertex, and no cycles are formed (Sipser 1997). In this case, the seed tile defined in the aTAM acts as the root of the tree, and each component represents a vertex in the undirected graph representing the topology of the target structure. The drawback of this technique is that designing spanning tress is an NP-complete problem.

2.3.2 Bottom-Up Self-Assembly Design

As the sophistication of self-assembling systems increase, it will be more challenging to use top-down design, as self-assembly is an algorithmically NP-complete problem. It will be difficult to use dissection techniques when the self-assembly process goes through multiple stages to create a target structure. Using an NP-complete problem to solve another NP-complete problem, i.e. using spanning trees to design self-assembling systems, is not computationally efficient.

Evolutionary computing is well-suited to solving NP-complete problems (Mitchell

1996), and offers the potential for a bottom-up self-assembly design approach to self-assembly. Evolved, self-assembled programs have been created using repositories of software components, as an example of software self-assembly (Li et al. 2008). The practical benefit of evolving software self-assembly over other automated programming methodologies, such as *Genetic Programming*, have not been demonstrated at the time of writing. Evolutionary computing has been applied in theoretical studies where chemical compounds were evolved (Buchanan et al. 2008), and where different co-operative bonding mechanisms were evolved in the context of the aTAM to create a single 10×10 square target structure (Terrazas et al. 2007). Evolutionary computing has been applied to solving the MTSP (*section 2.2.5: The Complexity of Self-Assembly*) using both theoretical 2D and 3D tiles, and also considering component rotations in both 2D and 3D (Vieira & Barbosa 2011). However, only square and cube target structures were considered, and no physical systems of the generated designs were implemented.

S-bots have been used to show a physical example of using evolutionary computing for bottom-up self-assembly design (Ampatzis et al. 2009). In this case, the communication sequence between two s-bots was evolved to enable their self-assembly. During the evolved communication sequence, the two s-bots determine which one is designated the seed component, and the other one self-assembles to it. Evolutionary computing has also been used to evolve the self-assembly sequence of cube-based robots, resulting in self-replication (Zykov et al. 2007). Although evolutionary computing has been applied to virtual systems, and communication between robotic units in physical systems, it has not been applied to generate component sets (including component morphology) in physical self-assembling systems.

2.4 Summary

This chapter critically reviewed the relevant self-assembly research pertaining to this thesis, by surveying physical self-assembling systems across physical scales, examining the computational aspects of self-assembly from the perspective of DNA computing, and contrasting top-down versus bottom-up self-assembly design methods in the literature. A summary of these three aspects of the self-assembly literature, including the shortcomings which the work in this thesis addresses, is provided below.

Physical Self-Assembling Systems

Artificial, physical self-assembling systems (including examples of static, dynamic, and templated self-assembly, and netted systems) from the macroscale, across the mesoscale, and down to the nanoscale were surveyed, in 1D, 2D, and 3D. There has been advancement in methods to reduce or prevent component interactions in static self-assembling systems, using shape and oligonucleotides in DNA-based components at the nanoscale for example. However, these examples and other systems in the literature do not demonstrate how component characteristics can be used to reduce or prevent errors while simultaneously be used to enable the self-assembly of closed target structures with symmetric/asymmetric features. Furthermore, these error reduction/prevention techniques have not been extended to 3D components, nor has component physical information to enable 3D component interactions (e.g. rotational properties) been considered at of the time of writing. In this thesis, evidence from physical experiments is provided to demonstrate how component characteristics in 2D and 3D can be exploited to reduce or prevent component interaction errors while simultaneously being used to self-assemble closed target structures with symmetric/asymmetric features.

Physical Information Encoding and DNA Computing

The role of asymmetry was discussed in both defining the notion of physical information (the generation of physical processes, such as self-assembly) and mathematical tiling, providing a link between self-assembly and models of pseudo-crystalline growth, such as the aTAM. As with all its extensions, the aTAM does not permit component rotations. It was demonstrated that although the incorporation of rotations may not have a significant reduction in terms of the number of components required in a system from an algorithmic complexity perspective, the inclusion of rotations is beneficial in exploiting symmetry in a target structure and parallelism in the self-assembly process. Although the aTAM was proven to be Turing universal at temperature two, the rgTAM proved that Turing universal computation is possible at temperature one in 2D, and similarly in 3D. The creation of temperature one systems are of significance, as temperature two co-operative bonding is difficult to implement in practice. As well, it was shown how the aTAM was used to prove that designing self-assembling system is an NP-complete problem. The work described in this thesis (resulting in physical systems) uses tile-based modelling, incorporating 2D and 3D rotations at temperature one, to address the complexity of designing self-assembling systems.

Designing Self-Assembling Systems

Primarily top-down design methodologies have been used to design physical self-assembling systems. It will be more difficult to use top-down design (such as dissection techniques) as the sophistication of these systems increases (e.g. large numbers of components interacting with one another and their environment in parallel, and when the self-assembly process goes through multiple stages to create a target structure). The development of a design methodology that is inherently bottom-up and coupled with bottom-up construction (self-assembly), and which encodes physical information to enable the self-assembly process is the principle contribution of this thesis. Furthermore, it is shown how this design process can be extended to incorporate evolutionary computing to address self-assembly being an NP-complete problem, and to incorporate staging where component physical information is leveraged to prevent and reduce errors to create more complex target structure that would not otherwise be possible.

Chapter 3

Physical Information Analysis

The work presented in this chapter was part of an initial investigation in the development of a self-assembly design methodology. Designing self-assembling systems is an algorithmically NP-complete problem (*section 2.2.5: The Complexity of Self-Assembly* and *section 2.3: Designing Self-Assembling Systems*). Based on the relevant literature (*Chapter 2: Literature Review*), an overarching self-assembly design methodology to address this design challenge would be one that is inherently bottom-up and based on the interplay between information and the generation of a process, incorporating the connection between self-assembly and computation (Winfree 1995). It is desired that such a self-assembly design methodology have a way to: (1) describe the information in a system, (2) model that information using software as a computationally efficient method to determine the outcome of an emergent system, and (3) upon successful evaluation in simulation, translate that information into a physical self-assembling system. Therefore, the purpose of this initial investigation was to analyze physical information in the context of designing self-assembling systems, and was accomplished by contrasting two systems.

The first system is an idealized model of self-assembly, where homogeneous components use localized, intra-component communication to create closed self-assembled structures with symmetric/asymmetric features (Bhalla & Jacob 2007). The second system is the original artificial physical self-assembling system, using two mechanical, passive components (Bhalla & Bentley in print), which is remarkable in its ability to create self-reproducing structures. These two systems were selected to investigate features of self-assembly where there are few practical physical constraints in the idealized model, and to contrast those features to ones in the pioneering physical system which is highly constrained. A summary of these features is provided at the end of this chapter, including: components, environment, initial conditions, energy, communication, conditional behaviour, positional information, target structures, and generated process. A method of applying a description of information is applied to both systems. The resulting self-assembly design methodology from this initial investigation is presented in *Chapter 4: The Three-Level Approach to Self-Assembly Design.*

3.1 An Idealized Self-Assembly Model

Here, self-assembly principles and mechanisms are abstracted to form a framework, based on the artificial systems presented in *section 2.1: Physical Self-Assembling Systems* and natural systems (Ball 1999, 2009a,c,d; Thompson 1917), including snow crystals and bacterial flagellum (*Chapter 1: Introduction*). This framework is the basis to the design of a virtual system. The purpose of this system is to investigate what information is required to create self-assembling systems in 2D and 3D with symmetric/asymmetric features using unit spherical components, by leveraging different patterns (based on the spatial arrangements of components occurring due to their assembly locations, e.g. forming square or hexagonal lattice patterns). Five prototypes are discussed to demonstrate how this framework can be leveraged to analyze and create self-assembling systems.

3.1.1 Framework

The mechanisms and constraints in natural self-assembly and netted systems do not need to be viewed separately. For the purposes of analyzing and creating self-assembling systems, these mechanisms and constraints can be abstracted to eight items as an expansion of the framework presented in (Bhalla & Jacob 2007):

- Components
- Environment
- Energy
- Assembly Protocol
- Spatial Relationship
- Localized Communication
- Rule Set
- Time

Components are defined by their properties. Such properties include, but are not limited to, shape, mass, scale, and material properties. Component properties can act as information encodings, enabling the self-assembly process.

The physical and chemical properties of the environment will influence the manner in which components interact with one another, as well as the way in which components self-assemble. For example, the environment medium can influence the motion of the components. The environment can provide various functionalities, such as a boundary to which components are confined to. The state of the environment, whether it is constant or fluctuating is an important factor in self-assembling systems. Controlling the parameters of a fluctuating environment can be used to enable the self-assembly process of components that have limited properties or constrained interactions.

In order for components to self-assemble, they need to be mobile in their environment. This requires components to have energy. Energy could be transferred to components from their environment and/or be available internally. Static self-assembling systems lead to structures or patterns in local or global equilibrium and do not dissipate energy.

Whereas dynamic self-assembling systems lead to structures or patterns that can only occur when the system is dissipating energy.

An assembly protocol defines the methods in which components can self-assemble. These methods are highly dependent on the scale of a system, as well as the physical and chemical properties of the components and the environment. For example, Watson-Crick complementarity can be used with DNA-based components at the nanoscale, hydrophobic/hydrophilic and capillary forces can be used at the mesoscale, and magnetism can be used at the macroscale.

The spatial relationship between the components and/or elements in their environment defines the underlying pattern formations capable by a system. Pattern formation has a great influence on the range of achievable self-assembled structures by a system, and is seen at all scales. The influence of a spatial relationship is seen in both natural systems (e.g. the hexagonal symmetry of ice crystals) and synthetic systems (e.g. the linking points of robotic units in modular and swarm robotics).

Localized communication is an important consideration in self-assembly. This is the primary factor in viewing self-assembly as an emergent property of decentralized systems. This is seen throughout nature at all scales. Localized communication dictates how components interact with one another and their environment. It can be viewed as a physical/chemical encoding (Bhalla & Bentley 2006), achieved through data, or other communication means.

The rule set can also be viewed as a physical/chemical encoding (Bhalla & Bentley 2006), or achieved through data communication. The rule set can be basic and still lead to a wide variety of self-assembled forms (e.g. the various supersaturation, temperature, and background gas settings lead to varying snow crystal morphologies). Or, the rule set can be internalized in the components and lead to a wide variation of self-assembled entities displaying many forms and performing many functions (e.g. DNA within cells).

Time can be used to specify the duration of the self-assembly process, and can also be divided into stages where components can be added to, or removed from, an environment. The environmental conditions, such as temperature, can also vary over time.

This proposed framework facilitates the melding of the various self-assembly principles and mechanisms across disciplines. This interdisciplinary view is beneficial in that it aids in the pursuit of creating artificial systems, mirroring the robustness of bottom-up construction, self-assembly, in nature.

3.1.2 Idealized Model Design

Based upon the above framework, an idealized model was created as a proof-of-concept method to analyze and create self-assembling systems. The objective of the idealized model was to determine how homogeneous, spherical components could self-assemble into 2D and 3D closed target structures with symmetric/asymmetric features.

Single components move around freely within the environment. One component is selected at random and placed in the environment, and set to remain stationary. This component acts as the seed particle to initiate the self-assembly process. When free components collide with stationary components, they either reflect or assemble with them and become incorporated into the aggregate structure as stationary components. The locations to which new free components are allowed to attach is determined through communication between neighbouring stationary components. Localized, intra-component communication was anticipated to allow for the creation of closed target structures, instead of resulting in open target structures, as created in diffusion-limited aggregation models (Witten & Sander 1981). Details of the proposed idealized model are as follows.

Spherical components in the system are all of unit radius and unit mass. The components are considered as solid entities and follow the principles of elastic collisions. Components are confined to a spherical environment. The surface of the environment is also treated as a solid structure in order to serve as a boundary to confine the components. The radius of the environment is specified at the beginning, and then remains fixed for the duration of a simulation run. Components are assigned random velocities within a user specified range and move around the 3D environment. When free moving components collide and attach to stationary components, their velocity is set to zero, and they become stationary themselves.

For computational efficiency, the components' and environment's shape was chosen to be spherical. Computing collision detection and response between components and with the environment is much simpler for spheres, in contrast to other 3D forms (e.g. polyhedra). Component morphology can be used as a physical or geometric means of encoding information to enable the self-assembly process (Bhalla & Bentley 2006). Since the components in this system are all unit spheres, additional component attributes are required. These additional attributes are described in terms of the assembly protocol, spatial relationship, localized communication method, and rule set.

The assembly protocol in this idealized model is based on the concept of *stickiness*. When two components collide, and each is in a particular state, the two components stick together to form an aggregate structure. One factor affecting a component's state is its location in the aggregate structure. A component's location in the aggregate structure is determined by the spatial relationship and localized communication between components.

The underlying pattern formation, or spatial relationship, between components defines the set of target structure morphologies. Locations on a component, referred to as *sticky sites*, determine the morphologies achievable by the aggregate structure. The user defines these sticky sites, on the 3D surface of the components. The locations of the sticky sites are the same for every component present in the system. Figure 3.1 shows two example components with different sticky sites and their resulting pattern formations.

For computational reasons, sticky sites are defined in a pairwise fashion. In Fig-

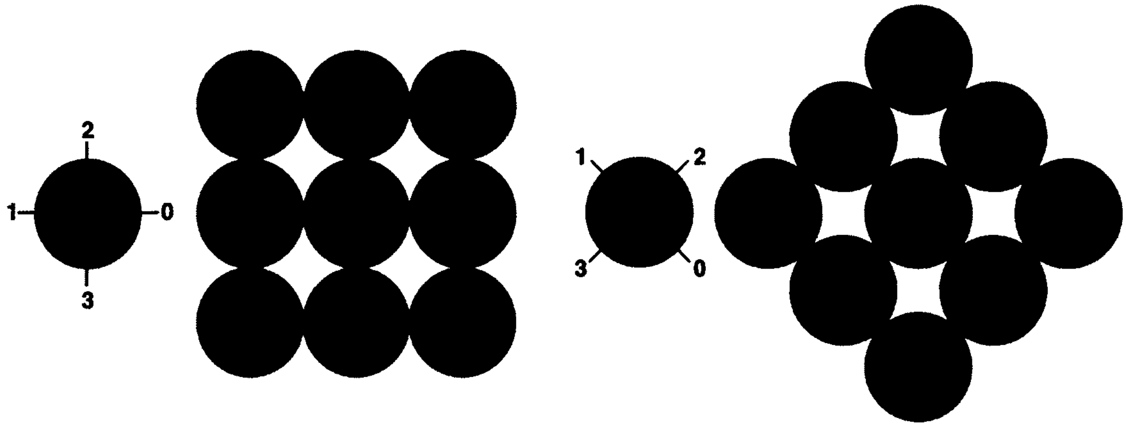


Figure 3.1: 2D square lattice pattern (left) and diamond lattice pattern (right).

ure 3.1, the pairs of sticky sites are the ones labelled (0,1) and (2,3). When a component collides with a stationary component, it attaches to the closest available sticky site on the stationary component. If no sticky site is available, the colliding component reflects off the stationary component. The manner in which two components stick together is determined by the pairwise relationship of their sticky sites. For example, if the available sticky site on the stationary component is at position zero, then the colliding component will attach via its sticky site at position one. This eliminates the need to calculate the orientation of each component. It is also used to allow for two forms of localized communication.

Components are able to communicate with their local neighbours. The user-defined spatial relationship determines the neighbourhood relationship of the components. In this idealized model, components can only communicate with their immediate neighbours, i.e. the ones they are directly connected to.

Communication between components is used to update two types of information. These two types of information are encoded into every component present in the system. Both types are used as a means for the components to infer geometric information from their neighbours in regards to their position in the aggregate structure.

This information is based on the axes that are created through the pairwise relationship of the sticky sites. The number of axes present determines the number of values used to represent this information. One axis is created for each sticky site pair. This can be viewed as x-y and x-y-z Cartesian coordinates for a 2D square lattice and a 3D cubic lattice respectively.

The first type of information is referred to as global axes information. Each component at the beginning of a simulation run starts with their global axes information set to (0,0) for the square lattice and to (0,0,0) for the cubic lattice examples. Each number represents a components location in relation to the x-y or x-y-z axis. Furthermore, the pairwise relationship of a sticky site set can be interpreted as the resulting axis having a positive and negative direction. The selection of which sticky site within a pair is negative or positive is arbitrary; its selection only needs to be consistently referred to in the manner in which it is selected.

When a component attaches to another component that is part of a self-assembled aggregate structure, it updates its global axes information. It does this by first requesting the global axes information from the component it has attached to. The component then updates its own global axes information by first incrementing or decrementing the value corresponding to the axes through which it is attached. Secondly, it either copies, increments, or decrements the rest of the values appropriately. The increment or decrement value is dependent on the type of spatial relationship used. For example, an increment and decrement value of one is used for square lattices (Figure 3.2).

The second type of information is referred to as axes count information. Initially, all components start with their axes count values set to (0,0,0,0) for the 2D and to (0,0,0,0,0,0) for the 3D Cartesian coordinates examples. In this case, two values are used to represent a components location in relation to each axis. Axes are used in a way to

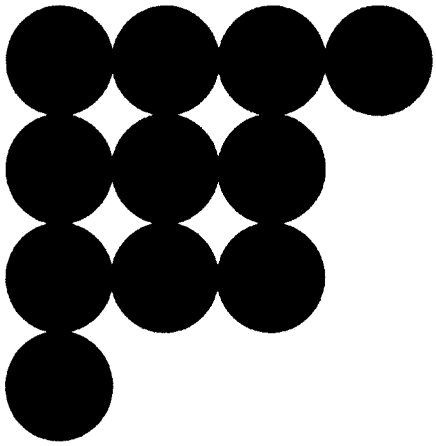


Figure 3.2: Global axes information for 2D square lattice pattern, with component (0,0) positioned at the centre of the grid.

allow components to infer the number of components that are located in both directions of each axis present. For a specific component that is part of an aggregate structure, this is the number of components to its left and to its right, for example.

The axes count information is updated when a component attaches to the aggregate structure. However, since a newly attaching component can affect the axes count information of other components in the self-assembled aggregate structure, multiple components need to update this information when a new component attaches in this scenario. Two cases arise for this type of information. The first case is when a component attaches to the aggregate structure, and it only has one neighbour. In this case, the axes count information only needs to be updated for one axis in one direction. Figure 3.3 shows an example of this case. The second case is when a component attaches to the aggregate structure, and it has more than one neighbour. In this case, axes count information is updated for multiple axes and/or in both directions of an axis. Figure 3.3 also shows an example of this second case. For both cases, updated information and communication is done locally as a propagation of information. This maintains the decentralized characteristic of natural self-assembling systems.

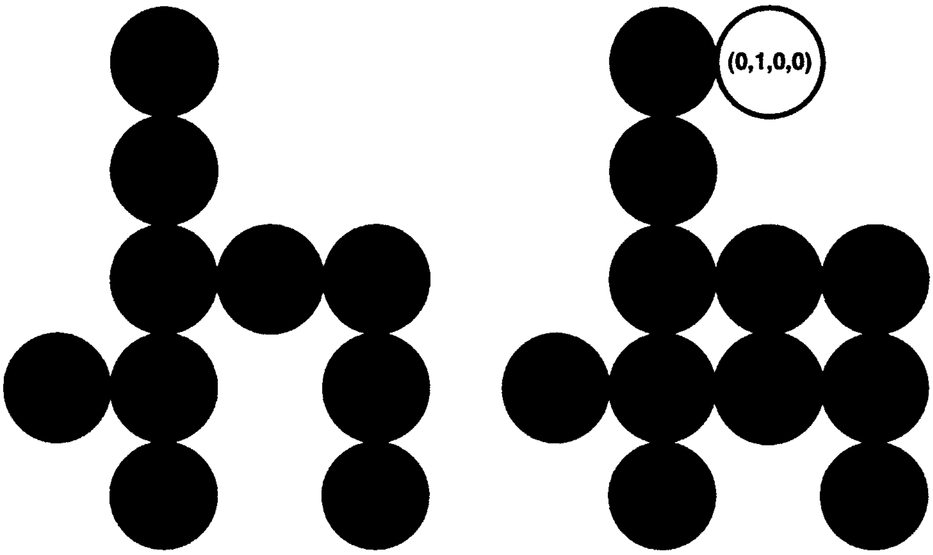


Figure 3.3: Axes count information (original configuration, left), with case one indicated with the black-outlined white-filed component, and case two indicated with the black-outlined grey-filled component.

These two types of information were incorporated into a rule set. By doing so, it was anticipated that the idealized model would be able to achieve the objective of designing 2D and 3D closed self-assembled structures with symmetric/asymmetric features. The rule set, which is specified by the user, is the same for all components that are present during a simulation run. A number and a type is used to reference rules. Rule type one is in reference to the global axes information, and rule type two is in reference to the axes count information. When a component attaches to the aggregate structure, it goes through each rule in the set and executes each rule that applies.

Each rule is defined as a state-action pair. The state comprises of two parts. The first part depends on whether the component is mobile or stationary. The second part is in reference to either of the two types of information. If both parts are satisfied, then an action is performed. The action in this case is the activation or deactivation of sticky sites. By doing so, it allows for the emergence of closed self-assembled structures.

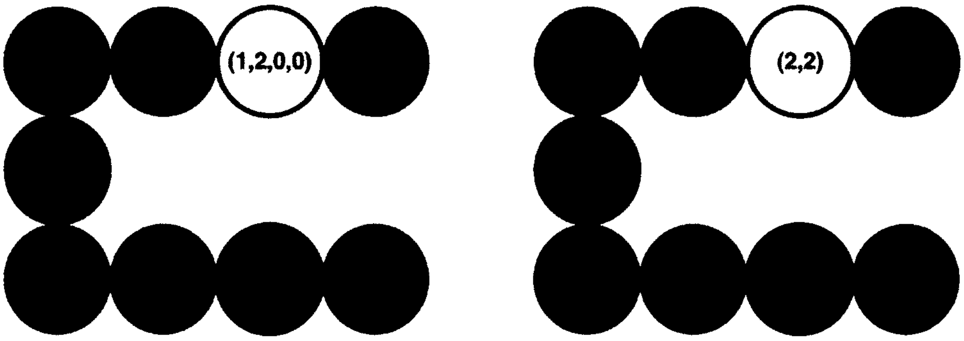


Figure 3.4: The two black-outlined components with the same axes count information (left); and the same two with unique global axes information (right).

Both rule types allow for the emergence of symmetric structures. However, the axes count information could possibly be the same for multiple components in the aggregate structure. This gives the potential for using one rule that is applicable to more than one component in the aggregate structure. Figure 3.4 (left) shows an example of this condition. In contrast, only the global axes information allows for the emergence of asymmetric structures. This is achieved because components can be identified uniquely through local communication, and thus a rule can be specified for a component in a unique state. Figure 3.4 (right) also shows an example of this condition.

Table 3.1 provides a summary of the idealized model in the context of the framework presented. The objective of the idealized model is to determine how spherical components can self-assemble into 2D and 3D target structures with symmetric/asymmetric features.

3.1.3 Prototypes

The software was written using BREVE (Klein 2002), a software package for visualizing decentralized multi-agent systems. To test the conceptual foundation of the proposed system, four target structures were specified: a cube, the letters N and B, a chair, and a mug (Figure 3.5).

Components	spheres of unit radius and unit mass
Environment	spherical boundary to contain components
Energy	components are set to a random velocity
Assembly Protocol	stickiness
Spatial Relationship	pairwise relationship of sticky sites
Localized Communication	global axes and axes count information
Rule Set	state-action pair (state: mobile vs. stationary, and global axes information vs. axes count information; action: activation/deactivation of sticky sites)
Time	duration to complete the self-assembly process

Table 3.1: Idealized model summary

Figure 3.5: Four target structures: cube (top-left), chair (top-right), NB (bottom-left), and mug (bottom-right); © 2007 IEEE.

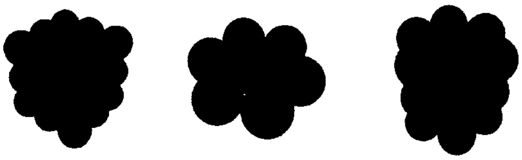


Figure 3.6: 3D cubic lattice (left), 2D hexagonal lattice (centre), and 3D layered hexagonal lattice (right); © 2007 IEEE.

These closed target structures were chosen because they have symmetric/asymmetric features. The morphologies of these target structures cannot be created through pattern formation exclusively. They require additional information, and therefore were appropriate candidates to test the capabilities of the idealized model.

In order to create these structures, three spatial relationships were needed. These three patterns are a 3D cubic lattice, 2D hexagonal lattice, and a 3D layered hexagonal lattice (Figure 3.6). For the 3D cubic lattice, the update scheme used for the global axes information was the one as described in the previous section. An increment or decrement of one was used depending on the positive or negative position of the axis the new component was self-assembling on, and all other values were copied.

The update scheme developed for hexagonal lattices was developed from the idea of fitting a hexagonal grid to a square grid. With this idea, three cases arise for updating the global axes information. Each case is described in reference to the sticky sites positions shown in Figure 3.7. When a free component attaches to a stationary component at either position four or five, an increment or decrement of two is used to update the vertical information. All other information is copied. The second case arises when a free component attaches to a stationary component in either the zero, one, two, or three position. This is a special case, and both the (0,1) and the (2,3) axes are considered as horizontal information. As a simplification of the resulting geometry, for each component the values for these two axes are the same. Hence, the vertical information correspond-

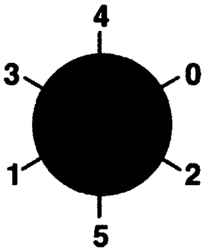


Figure 3.7: Sticky site locations for the 3D hexagonal grid, with positions 6 and 7 located at the centre-front and centre-back of the component indicated by the black circle.

ing to axis (4,5) is incremented or decremented by one, and the horizontal information corresponding to (0,1) and (2,3) is incremented or decremented by one, depending on the resulting position of the free component in relation to the quadrant on a Cartesian plane. In the 3D case, an update to the height information corresponding to the (6,7) axis is incremented or decremented by one, and all other information is copied. As a result, this update scheme for the global axes information allows each component in the aggregate structure to have a uniquely identified position. Figure 3.8 gives an example of this idea for the 2D case.

The 2D hexagonal and 3D layered hexagonal lattices were used to create the letters N and B, and the mug, respectively. The 3D cubic lattice was used to create the cube and the chair target structures. These spatial relationships were used in five prototype systems. The first four prototypes corresponded to the four target structures. For these prototypes, rules utilizing the global axes information were used. This was done because it was a simplified approach to test the conceptual basis of the idealized model, since these prototypes are a proof-of-concept. The parameter settings corresponding to these four prototypes are summarized in Table 3.2 (page 67). The number of rules for each prototype equals the number of components needed to create its respective target structure. A fifth prototype was constructed that utilizes rules based on both the global axes information

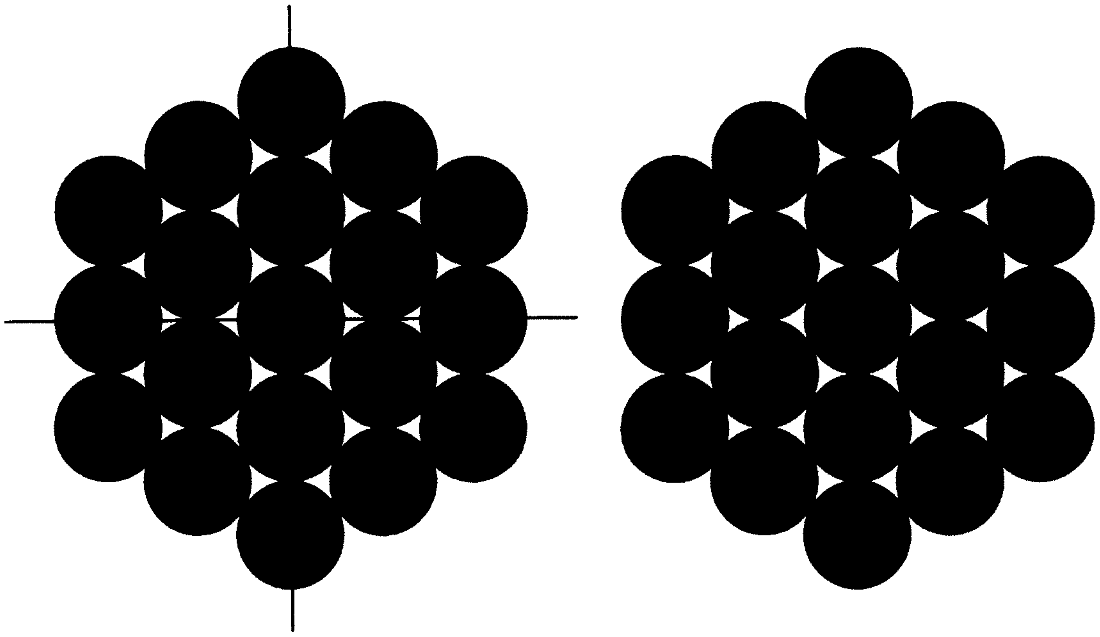


Figure 3.8: Global axes information for a 2D hexagonal lattice shown as a 2D square grid (left), and the global axes information values (right).

and the axes count information. This was done as an attempt to reduce the number of rules required by taking advantage of the symmetric properties of a target structure. This experiment was done to create the cube target structure. The parameter settings corresponding to this prototype are also included in Table 3.2.

3.1.4 Prototype Results

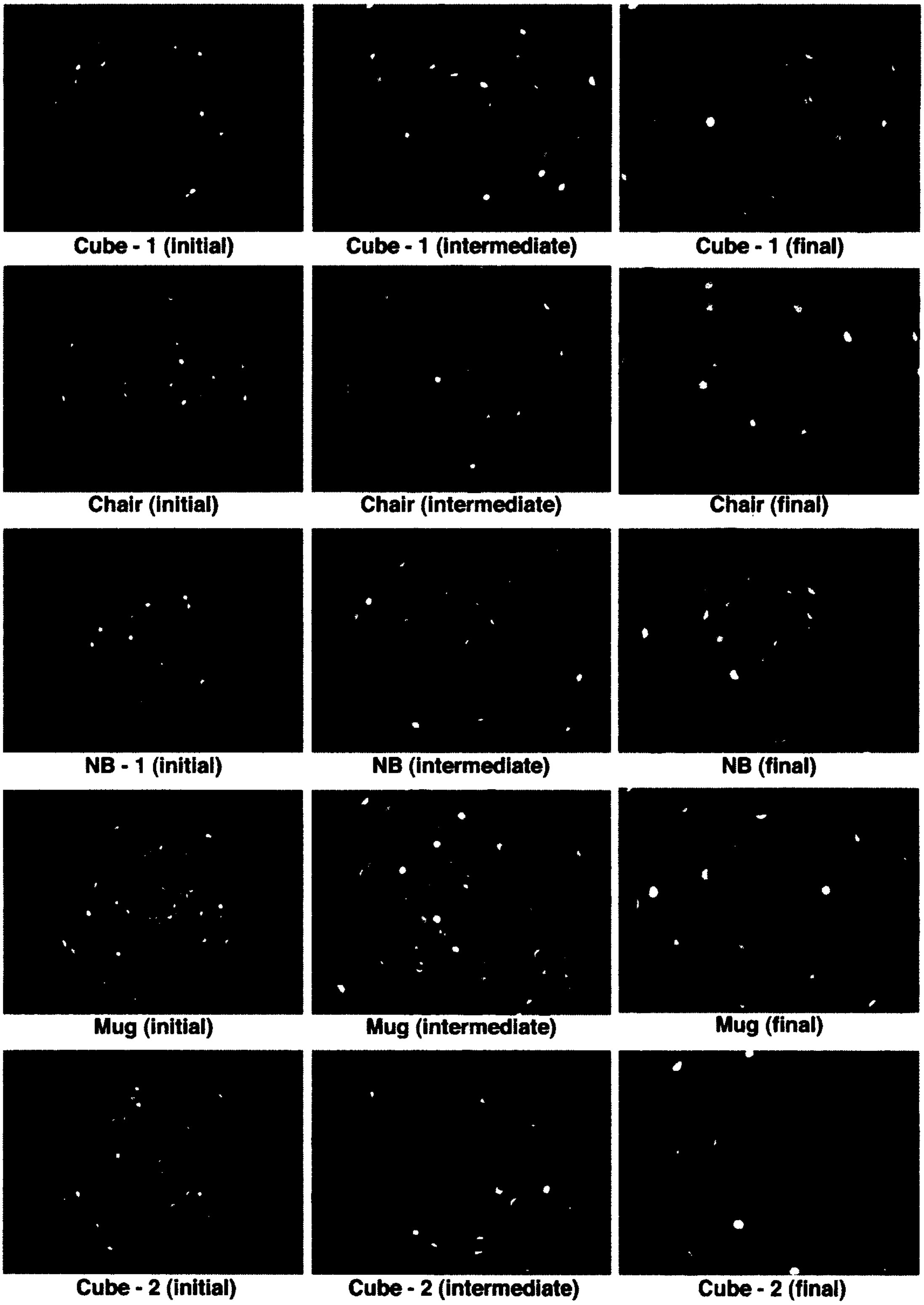
Each of the five prototypes was successful in having a subset of the overall components self-assemble into their respective closed target structure. The robustness of the bottom-up construction process of self-assembly was demonstrated in each of these five prototypes. Figure 3.9 shows the results of the first four prototypes, where components were assigned random colours. Each time a particular system was executed, the self-assembly process was unique, but the final closed target structure was always achieved. For example, in one simulation run, the base of the mug self-assembles, followed by the cylinder,

[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Cube - 1	150	10	30	GA	44	44	S
Chair	150	10	30	GA	40	40	S & A
NB	150	10	30	GA	36	36	A
Mug	300	10	30	GA	87	87	S & A
Cube - 2	150	10	30	GA & AC	35	44	S

Table 3.2: Parameter settings and target structure features for the five prototypes (P): number of simulation components (#SC), component velocity (CV), environment radius (ER), rule type (RT) of either global axes (GA) and/or axes count (AC) information, number of rules (#R), number of components (#C), and symmetric/asymmetric (S/A) features

and finally followed by the handle. In another simulation run, each of the three main features of the mug could be partially self-assembled at some time point, and continue to self-assemble in parallel, as shown in Figure 3.10 (page 69). This was achieved because the actions associated with the rules specified all the specific sticky locations that should be activated or deactivated for each component comprising the desired entity.

Another qualitative observation from running these four prototypes showed that even the mug (which consists of approximately twice as many components as the other three target structures) was able to self-assemble easily. This was due to the fact that most of the components present in the mug are part of the cylinder. As a result, there are a greater number of locations and a much larger surface area (in comparison to the other three closed target structures) where free components can self-assemble to. For example, the legs of the chair were on occasion difficult to self-assemble, because free components would take a long time to collide with the stationary components comprising the legs,



Cube - 1 (initial)

Cube - 1 (intermediate)

Cube - 1 (final)

Chair (initial)

Chair (intermediate)

Chair (final)

NB - 1 (initial)

NB (intermediate)

NB (final)

Mug (initial)

Mug (intermediate)

Mug (final)

Cube - 2 (initial)

Cube - 2 (intermediate)

Cube - 2 (final)

Figure 3.9: Five prototype results; © 2007 IEEE.

Figure 3.10: Four examples of different mug configurations, demonstrating the bottom-up parallel construction characteristic of self-assembly; © 2007 IEEE.

due to the stochastic behaviour of the system.

The above observations were also present in the fifth prototype. In this prototype, the use of rules utilizing the axes count information reduced the number of rules needed to create the cube target structure. This is an important result, because it shows that there is potential in creating systems that are scalable, as the number of components comprising the desired entity increases. Figure 3.9 also shows the results of this experiment (Cube 2). Components present in the bottom and top two square elements change their respective colours to red, as an indication of the execution of rules associated with the global axes information. The components present in the vertical structures change their respective colours to black. This indicates the execution of rules associated with the axes count rules. Only three rules were needed to define the twelve components present in the vertical elements of the cube target structure.

Exploiting the parallel features in a target structure is one method to reduce the number of rules required for scalability purposes. Since the planar surfaces present in each of the four target structures is either very small or essentially non-existent, the spatial relationship between components could not be exploited as a method to reduce the number of rules. For example, a system utilizing a 2D square lattice spatial relationship in order to create a large 2D filled square only needs rules to specify the outer perimeter or boundary of the desired filled square. The components in the interior of the square can self-assemble purely based on pattern formation in this idealized model. Further methods to exploit the spatial relationship of a system could be used to reduce the number of required rules.

The desired entities in the five prototypes would each self-assemble approximately between forty and three hundred seconds. The focus in these experiments was not on construction time and instead on demonstrating that the self-assembly of closed target structures with symmetric/asymmetric features was feasible. However in the future, creating systems with predictable time frames is important in realizing new self-assembling technologies. The combination of compact rule sets, exploiting the spatial relationship of a system, and more advanced assembly protocols (e.g. swarm intelligence) should be explored to create systems that can self-assemble in more predictable time frames.

3.2 A Self-Reproducing Analogue

In contrast to the virtual system using unit spherical components, the first mechanical analogue to natural self-assembly by L.S. Penrose and R. Penrose is used to demonstrate how it is possible to design a self-assembling system via physically encoded information. An overview of their system is provided first. Next, a description of how the physically encoded information in the components, and their environment, interacts through a set of rules. This is followed by specifications and by test results from an implementation created through reverse engineering to verify their self-reproducing analogue.

3.2.1 System Overview

Self-reproduction was once thought to be an intrinsic property of nature, and one that could not be mimicked in artificial systems. L.S. Penrose and R. Penrose were the first to show a mechanical self-reproducing analogue (1957). Their system is akin to molecular amplification in the form of templated self-assembly. They created two component types,

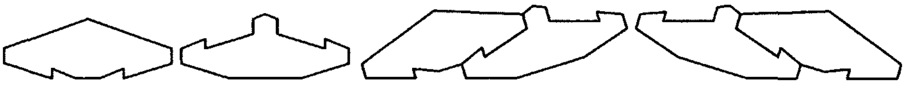


Figure 3.11: From left to right: component α and β (in their neutral positions), and complex $\alpha\beta$ and $\beta\alpha$.

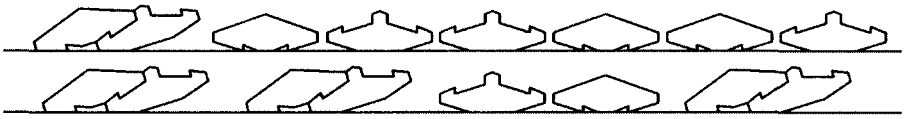


Figure 3.12: Example scenario with initial configuration (top) and final configuration (bottom).

labelled here as α and β. These two components connected in either an $\alpha\beta$ or $\beta\alpha$ configuration (Figure 3.11). Multiples of these α and β components were confined to a track in a random ordering. The track was shaken horizontally, in 1D, which allowed components to interact with one another and their environment. If only individual α and β components were shaken, self-assembly would not occur. However, if an $\alpha\beta$ (or a $\beta\alpha$) seed complex was placed on the track, only then would neighbouring α and β (or β and α) components self-assemble into $\alpha\beta$ (or $\beta\alpha$) complexes respectively (Figure 3.12).

3.2.2 Rules

Although L.S. Penrose and R. Penrose did not discuss this in particular, they were able to achieve artificial self-reproduction through the morphology of each component type and the design of the environment. There are ten information locations physically encoded through hooks, latches, and neutral sites (where no assembly occurs) on the two component types (Figure 3.13).

There are ten component interaction rules present. The information associated with each of these rules is physically encoded by shape. Rules (3.1) and (3.2) directly dic-

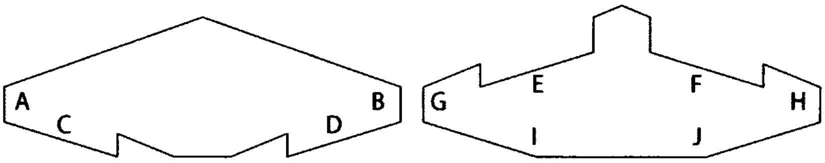


Figure 3.13: α (left) and β (right) identifying regions of physically encoded information using shapes.

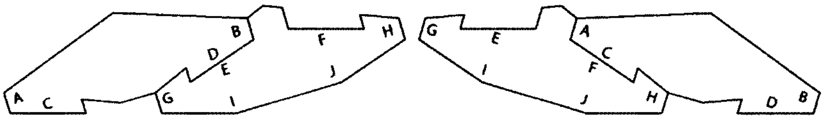


Figure 3.14: $\alpha\beta$ assembly (left, using rule (3.1)) and $\beta\alpha$ assembly (right, using rule (3.2)).

tate the self-assembly of the two component types, by hooking-and-latching together (Figure 3.14).

$$D\ fits\ E \to\ D +\ E \tag{3.1}$$

$$C\ fits\ F \to\ C +\ F \tag{3.2}$$

There are also two sets of four rotation rules, clockwise (cw) and anti-clockwise (acw). The first set (3.3) to (3.6) applies to the creation of $\alpha\beta$ complexes (Figure 3.15). The second set (3.7) to (3.10) applies to the creation of $\beta\alpha$ complexes (Figure 3.16).

$$A\ rotates_{acw}\ H \tag{3.3}$$

$$J\ rotates_{acw}\ A \tag{3.4}$$

$$A\ rotates_{acw}\ B \tag{3.5}$$

$$J\ rotates_{acw}\ G \tag{3.6}$$

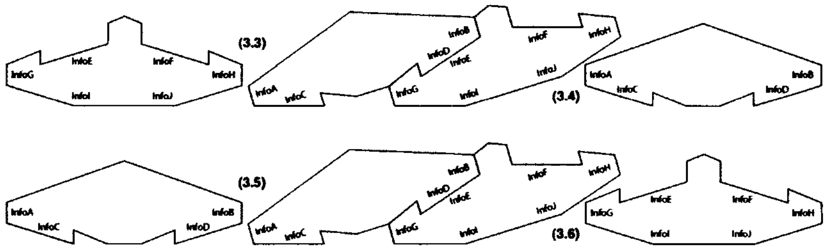


Figure 3.15: Rotation rules (3.3) to (3.6) promote the creation of new $\alpha\beta$ complexes.

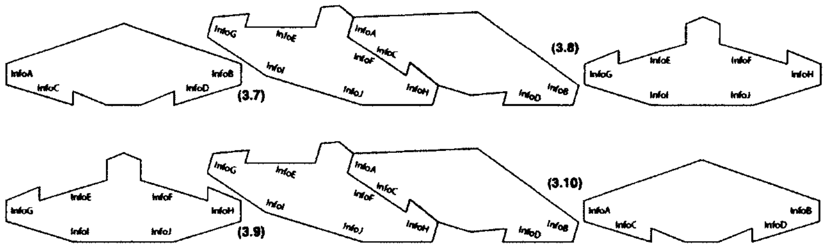


Figure 3.16: Rotation rules (3.7) to (3.10) promote the creation of new $\beta\alpha$ complexes.

$$I\ rotates_{cw}\ B \tag{3.7}$$

$$B\ rotates_{cw}\ G \tag{3.8}$$

$$I\ rotates_{cw}\ H \tag{3.9}$$

$$B\ rotates_{cw}\ A \tag{3.10}$$

To facilitate the component interaction rules (assembly and rotation), there are two environment rules in this system. The first environment rule is the prevention of components from *flipping* when rotating, so as not to move into a position where self-assembly cannot occur. The second environment rule is the provision of enough *rotational flexibility* for the components, and aids in bond creation by allowing the components to slide and hook-and-latch onto one another. These two environment rules result from having an appropriate environment height (Figure 3.17).

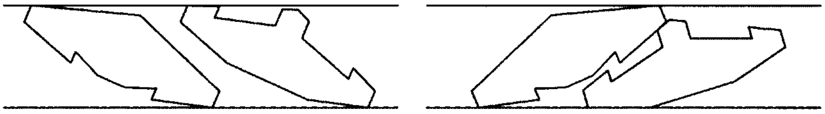


Figure 3.17: Environment height constraints (represented with lines) prevents components from flipping (left), and provides rotation flexibility to aid in bond creation between components (right).

In addition to the geometry of the environment, the shaking motion of the environment is horizontal (1D). This motion is applied to both individual components (initially placed in their respective neutral position) and seed complexes. These component interaction and environment rules, along with initially placing either an $\alpha\beta$ or $\beta\alpha$ seed complex, ensures which information comes in contact and the amplification of $\alpha\beta$ or $\beta\alpha$ complexes through two autocatalysis system rules (3.11) and (3.12). These two rules ensure that self-assembly only occurs in the presence of a seed complex (the ; symbol denotes separate entities). When the environment is shaken horizontally, $\alpha\beta$ or $\beta\alpha$ complexes act as a catalyst as they are not consumed in the creation of creating $\alpha\beta$ or $\beta\alpha$ complexes using assembly rules (3.1) and (3.2) and rotation rules (3.3) to (3.10). Individual α and β components do not act as a catalysts in this system, since when the environment is shaken horizontally α and β components cannot rotate, and assembly rules (3.1) and (3.2) do not occur as a result.

$$\alpha\beta\ \mathit{autocatalysis}\ \alpha;\beta \rightarrow\ \alpha\beta;\ \alpha\beta \tag{3.11}$$

$$\beta\alpha\ \mathit{autocatalysis}\ \beta;\alpha \rightarrow\ \beta\alpha;\ \beta\alpha \tag{3.12}$$

3.2.3 Specifications

L.S. Penrose build upon this system to develop more sophisticated self-replicating machines in 2D (1958; 1959). This system was independently verified by Edward F. Moore

(1962). However, Moore did not provide an explanation to how this system works (referring to it as L.S. Penrose's basic model), nor did he provide any explanation to how he constructed the system. "No picture of Penrose's basic model is included in this paper, since if the reader attempts the problem of how to design the shapes of the units [α] and [β] so as to have the specified properties, the difficulties he will encounter in his attempts will cause him to more readily appreciate the ingenuity of Penrose's very simple solution to this problem" (Moore 1962).

In pursuit of reverse engineering this system, one can certainly attest to the difficulty in constructing it, and one does have a greater appreciation for this simple, yet powerful, pioneering self-assembling system. However, in addition to an explanation of how this system works in the previous section through a set of rules, the specifications are provided for how to construct this system.

The drawings provided in Penrose & Penrose (1957) were used to determine the specification of the α and β components. Mathematical specifications for the main surfaces of the two components is provided in *section A.1: Penrose Components.* The width of both components is 35 mm, and the height of α is 12 mm and β is 13 mm. The depth of the components was not specified in Penrose & Penrose (1957). Component depth is primarily dependent on the construction material, and will influence bond strength, as well as the dimensions of the environment.

An environment height of 17 mm was determined by reverse engineering to being capable of achieving the two environment rules. The width and depth of the environment were not provided in Penrose & Penrose (1957). The width primarily depends on the number of components, and the shaking strength used. The depth of the environment primarily depends on the depth of the components. A balance must be maintained to allow for the components to be able to slide back and forth, but not to be able to *twist* and not bond correctly. Also not discussed in Penrose & Penrose (1957), the shape of the

[redacted]

Figure 3.18: Machined component α (left) and β (right).

ends of the environment will play a role in the self-assembly process. The ends can either break, maintain, or enhance component bonding. Although there may be particular shapes to enhance the self-assembly process, vertical ends are sufficient.

The mathematical specifications of the components and the environment were used to create a set of physical specifications, accounting for tolerances. In the original system it was suggested that components could be made from wood or vulcanite (Penrose & Penrose 1957). In this implementation, α and β components were made from standard brass plate (7/64" depth; approximately 2.78 mm). The components were cut using a 3-axis milling machine (CNC Takumi V6). The vertical edges of the top point of the β component were slightly narrowed, by 2 microns. This provided enough tolerance for the two components to hook-and-latch together, and not disassemble easily. In addition, a slight curvature was put at eight points, related to four information locations on the two components. This curvature also assisted in creating and maintaining bonds between components. Figure 3.18 provides a photograph of the physically implemented α and β components. The environment for the components consisted of three main parts: back, middle, and front. These parts were made from clear acrylic sheet (3 mm depth), and cut on a laser cutter (Trotec Speedy 300 Laser Engraver). The three parts were connected to hold the components, using two stainless steel screws and wing nuts. Environment specifications for this system are provided in *section B.1: Penrose Environment.*

[redacted]	[redacted]
$\alpha\beta$ α β α β α β	$\alpha\beta$ $\alpha\beta$ $\alpha\beta$ $\alpha\beta$
[redacted]	[redacted]
$\beta\alpha$ β α β α β α	$\beta\alpha$ $\beta\alpha$ $\beta\alpha$ $\beta\alpha$
[redacted]	[redacted]
α β α β $\alpha\beta$ α β	$\alpha\beta$ $\alpha\beta$ $\alpha\beta$ $\alpha\beta$
[redacted]	[redacted]
$\beta\alpha$ β α α β α β	$\beta\alpha$ $\beta\alpha$ α $\beta\alpha$ α

Figure 3.19: Initial (left) and final (right) configurations after running each of the four system tests.

3.2.4 Test Results

Four tests were conducted on the reverse engineered physical system. Each test configuration was shaken horizontally by hand six times (three time to the left and three times to the right). Details of the test procedure is provided in *section C.1: Experimental Procedure for the Penrose System.* Figure 3.19 provides photographs of the test results. There are three characteristics of the implemented system that should be noted: (1) component-component bond strength, (2) component-environment spatial relationship, and (3) shaking motion of the environment.

As discussed in Frietas & Merkel (2004), the strength of the bonds are weak, as they are simply mechanical bonds. In the steps towards physically reconstructing this system, it was also observed that the mass of the components can influence the bond strength. Earlier prototypes were constructed using plastic components and aluminum components of the same specification. Brass components (which have a higher density) were found to be superior. Brass was used instead of increasing the thickness of the components. The weight of the components also helps prevent them from moving vertically in the environment when irregular shaking is applied. When this occurs, undesirable, spontaneous assembly can occur.

It was observed that two assembled components occupy less physical space as two α and β components in their neutral positions. The implication here is that as the self-assembly process progresses, more free space is created in the environment. As a corollary, the more free space there is, the more energy is required. It was also observed that the environment does not necessarily have to be shaken horizontally (1D). The environment can also be shaken in an up-and-down motion, e.g. +/- 45 degrees from its centre. A more in depth investigation into the affects of the environment in this system is required for future analysis. The successful results show that the original system by L.S. Penrose and R. Penrose truly is a mechanical analogue to self-reproduction.

3.3 Summary

This chapter investigated physical information analysis in the context of designing self-assembling systems by contrasting two systems, an idealized model and L.S. and R. Penrose's original self-reproducing analogue. It is anticipated that a design methodology to address the challenges of creating self-assembling systems, particularly self-assembly being an algorithmically NP-complete problem, would be one that is inherently bottom-up and based on the interplay between information and the generation of a process. These two systems were selected as an initial step to verify if such a design methodology was feasible. The two systems were used to test the three desired characteristics of such a self-assembly design methodology: (1) describe the information in a system, (2) model that information using software as a computationally efficient method to determine the outcome of an emergent system, and (3) upon successful evaluation in simulation, translate that information into a physical self-assembling system.

Table 3.3 provides a summary of the features of the idealized model and the original self-reproducing analogue. For both systems, a set of simple rules can be used to enable the self-assembly process. Since all the components were unit spheres in the idealized model, information in the form of two localized communication methods using message passing (global axes information and axes count information). These two forms of localized message passing created gradients within the emerging structure, similar to protein gradients in multicellular organisms (Wolpert 1998), were required to compensate for the lack of distinct component attributes. These two forms of information were able to create closed target structures with symmetric/asymmetric features. In contrast, the original self-reproducing analogue used components with unique physical features. The *shapes* of the components facilitated the encoding of component-to-component and component-to-environment interactions. The component and environment information, along with the set of interactions rules were used to create a successful replica of the original artificial self-assembling system. The simple hook-and-latch is an elegant method for mirroring the key-lock principle of enzyme action (*section 2.2.1: Physical Information Encoding in Nature*). However, these bonds in this case were relatively weak, and it would be difficult to create structures in 2D and 3D using only hook-and-latch bonds.

The ideal model provided a computationally efficient method to evaluate the outcome of a system. One advantage, as part of the initial investigation, is that it was possible to explore many forms of underlying component pattern formations. The disadvantage of the examples provided is that they relied on a seed particle, whereas an optional seed particle would allow for a greater variety of systems to be explored. However, using the other characteristics of the idealized model to extend a tile-based model such as the aTAM (*section 2.2.3: The abstract Tile Assembly Model*) would be help-

Components	unit spherical components with a common set of communication rules	α and β components
Environment	spherical boundary	linear boundary
Initial Conditions	seed component placed in the centre of the environment and other components randomly placed throughout the environment	seed complex placed anywhere in the environment and other components randomly placed throughout the environment in their neutral positions
Energy	internalized within the components resulting in movement in 2D and 3D	vibrational energy in 1D transfered from the environment to the components
Communication	localized message passing between components	component physical properties
Conditional Behaviour	state-action pair of global axes and axes count information resulting in component-to-component interactions	fits, rotates, and autocatalysis rules
Positional Information	components can infer their global position and relative position in the target structure	left and right positional information of the two components comprising the seed complex and resulting structures
Target Structures	variety of 2D and 3D structures with symmetric/ asymmetric features	either αβ or βα structures exclusively
Generated Process	self-assembly	self-assembly and self-reproduction

Table 3.3: Comparison of the features of the idealized model and the original Penrose self-reproducing analogue

ful in advancing the theoretical aspects of self-assembly research. These characteristics include modelling concurrent self-assembly (as multiple regions in the emerging structure could self-assemble simultaneous), using symmetry to reduce the number of unique rules/information (achieved through the axes count information), and exploiting asymmetry (achieved through global axes information). In an extended tile-based model not requiring a seed particle, concurrency could be realized using multiple emerging substructures and regions within substructures, symmetry could be realized through component rotation (*section 2.2.5: The Complexity of Self-Assembly*), and asymmetry could be realized by allowing components to have distinct information regions.

The original self-reproducing analogue demonstrated how a set of rules could be translated into a physical system. The use of neutral information (in both systems) enabled the self-assembly of closed target structures. However, it is difficult to create several shapes to implement hook-and-latch bonds to assemble many types of components together (fits rules). Furthermore, it is also difficult to implement error-correction mechanisms when using hook-and-latch bonds exclusively. Ideally, a method to encode several forms of component interactions, including rotations, that are applicable to 2D and 3D would advance the state of the art of physical self-assembling systems.

The success in analyzing these two system, along with identifying their advantages and disadvantages, supports the pursuit of developing the desired self-assembly design methodology. The three main attributes of these two investigations, (1) being able to describe a system through a set of rules, (2) modelling those rules to determine the outcome of a system in software, and (3) translating to a physical systems by mapping those rules using physically encoded information, is the basis to the self-assembly design methodology presented in the next chapter, *Chapter4: The Three-Level Approach to Self-Assembly Design.*

Chapter 4

The Three-Level Approach to Self-Assembly Design

In pursuit of answering the question of how one could specify a set of rules that can be mapped to a physical system using physically encoded information, the self-assembly of a variety of 2D and 3D closed target structures with symmetric/asymmetric features was considered. Just as L.S. Penrose and R. Penrose took a biomimetic approach (inspired by nucleic acid complexes in chromosomes), the self-assembly design methodology presented in this chapter is a biomimetic one, inspired by the central dogma of molecular biology (the transfer of genetic information to create proteins, physical shapes). The resulting design methodology, referred to as the *three-level approach* (Bhalla et al. 2007), is a combination of the idealized model and the first mechanical analogue of self-assembly.

The three phases to the three-level approach are: (1) definition of rule set, (2) virtual execution of rule set, and (3) physical realization of rule set (Figure 4.1). The motivation behind the three-level approach is in finding the fundamental information structures and rules that enable self-assembly in theory (level one), testing and refining those self-assembly rules through simulation (level two), and testing and refining those self-assembly rules through embodied physical experiments (level three). The three-level approach provides a bottom-up method for designing physical self-assembling systems. This is achieved by being able to directly map a set of rules to a physical system.

Level One: Definition of Rule Set The highest level, level one, is concerned with being able to describe any rule set. The rules could be of several types. For example, rules represented mathematically could be of the string re-writing form, e.g. L-Systems (Prusinkiewicz & Linenmayer 1990; Jacob 2001). The two types of rules presented in

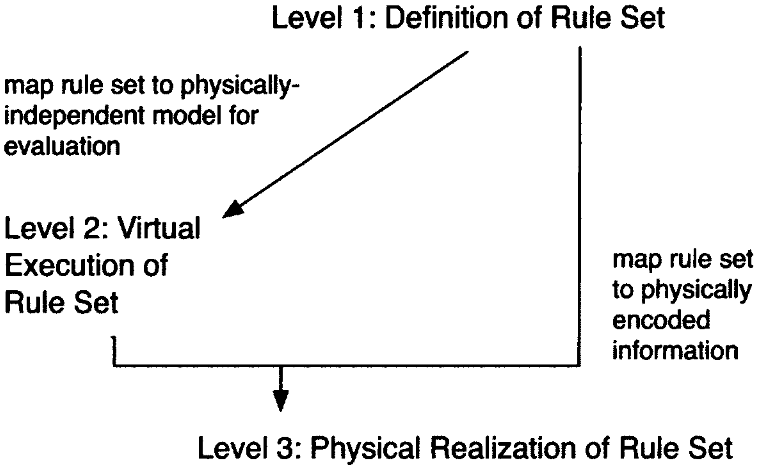

Level 1: Definition of Rule Set
map rule set to physically-independent model for evaluation
Level 2: Virtual Execution of Rule Set
map rule set to physically encoded information
Level 3: Physical Realization of Rule Set

Figure 4.1: The three-level approach to self-assembly design.

section 3.1 An Idealized Self-Assembly Model used state-actions pairs. As an extension to the rules presented in *section 3.2: A Self-Reproducing Analogue* that describe a system akin to molecular amplification, rules expressed abstractly could mirror those seen with proteins, e.g. A fits B, C+D fits E, F breaks G+H (where the letters represent physically and chemically encoded information). As well, rules in which physically and/or chemically encoded information is effected by natural phenomena should also be addressed. Such rules could include temperatureT breaks I+J. For example, the physical equivalent to this rule represents that DNA is denatured at a high enough temperature (i.e. double strand breaks apart into two single strands).

Level Two: Virtual Execution of Rule Set Execution of a set of level one rules is accomplished at the mid level, level two. Information comprising the components and their environment, as well as the rule set present, are expressed using modelling. This could be implemented through a simulation or a model (*section 3.1 An Idealized Self-Assembly Model*). In order to evaluate an emergent system, it has to be allowed to run (Wolfram 2002). This is the primary purpose of this level. The outcome of the simulation can determine if, in order for a self-assembled target structure to emerge, the level one rules or the information related to the different component types and their environment need to be modified. Representations used for the information and the rules should be independent of any physical medium. This has two benefits. The first is that it allows for the simulation to remain simple and computationally efficient to run (modelling the interactions of complex forms is a difficult task). The second is that it then allows for the integration of various materials and self-assembly mechanisms, which may be required depending on the target structures.

Level Three: Physical Realization of Rule Set At level three, information comprising the components and/or their environment, together with the level one rules present in the system, are mapped from their virtual representations to a physical system. Since this mapping is done directly, it is independent of the results observed from the level two modelling. The consequence of this is that mapping can occur before, during, or after a simulation run. *Section 3.2: A Self-Reproducing Analogue* shows how a set of rules can be mapped directly to create a physical self-assembling system.

The three-level approach provides a high-level description to designing a self-assembling system. The following level one rules, level two model, and level three physically encoded information were designed with the potential to create 2D and 3D self-assembled closed target structures with symmetric/asymmetric features to test the thesis hypothesis.

4.1 Level One: Definition of Rule Set

At level one, a self-assembling system is defined by three categories of rules: *component*, *environment*, and *system*. These rules are in the context of component movement, spatially in 2D and 3D. The objective of these rules is to provide a high-level, abstract description of the type of self-assembling systems used in this thesis.

Component Rules: specify component information. Conceptually similar to DNA tiles, components are either squares (2D) or cubes (3D). Each edge/face of a component serves as an *information location*, in either a four-point (Top, Left, Bottom, Right) or six-point (Top, Left, Bottom, Right, Front, Back) arrangement for 2D and 3D components respectively (Figure 4.2). Information is represented by a capital letter, A to H for 2D components and I to T for 3D components. A subscript (1 to 4) is used with each capital letter (e.g. N_4) to indicate orientation on a 3D component's face (Figure 4.2). The dash symbol ($-$) represents a neutral site (where no assembly information is present). The spatial relationship of a component's information defines its type (Figure 4.2).

Environment Rules: specify temperature (ϕ) and *boundary* constraints. An assembly protocol must at least meet the temperature for assembly bonds to occur. The boundary confines components to the environment. Components are permitted to translate and rotate in 2D and 3D systems. In addition, components have rotational information and can be reflected in 3D systems.

System Rules: specify component type frequency, and component-to-component information interactions (i.e. assembly interactions) and component-to-environment interactions (transfer of energy and boundary interactions). There are two types of system interaction rules referred to as *fits* and *breaks* rules. If two complementary pieces of information come into contact, (e.g. A fits B), it will cause them to assemble. This rule type is commutative (e.g. if A fits B, then B fits A). Furthermore, fits rules encapsulate component-to-component rotational interactions in 3D systems. A subscript (360, 180,

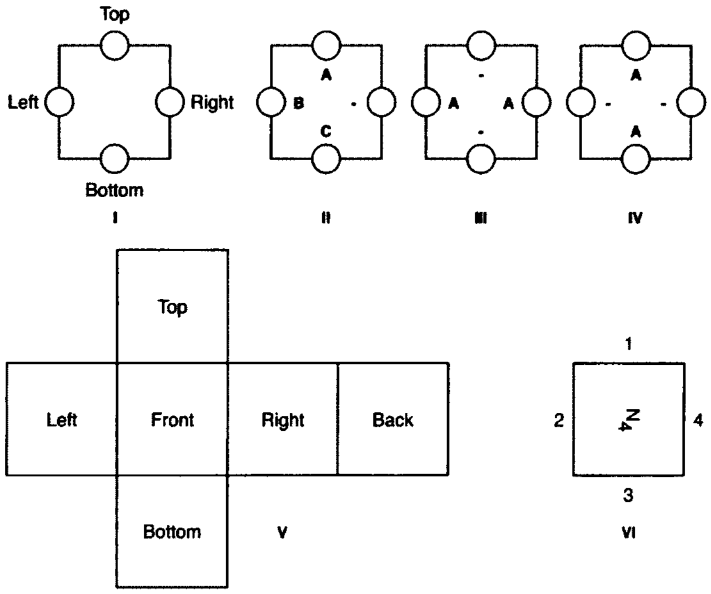


Figure 4.2: 2D and 3D component spatial information relationship (I and V respectively), an example of information orientation on a 3D component's face (VII), and example 2D component types (where III and IV are of the same type under planar rotation and are distinguished from II).

and 90) is used to represent if the faces of complementary 3D components can fit together in four, two, or in one way respectively (e.g. M fits_{180} N). If two assembled pieces of information experience a temperature of two (ϕ_2), then their assembly breaks. The interaction rules used in the self-assembling systems presented in this work are provided in Table 4.1 (2D) and Table 4.2 (3D).

4.2 Level Two: Virtual Execution of Rule Set

At level two, a rule set is mapped to an abstract model for efficient evaluation, and is used to determine if physical evaluation of a rule set is applicable at level three. In the original publication of the three-level approach an agent-based model (similar to the model presented in *section 3.1: An Idealized Self-Assembly Model*) was used at level two

A fits B → A+B	ϕ_2 breaks A+B → A ; B
C fits D → C+D	ϕ_2 breaks C+D → C ; D
E fits F → E+F	ϕ_2 breaks E+F → E ; F
G fits H → G+H	ϕ_2 breaks G+H → G ; H

Table 4.1: 2D interaction rules (fits and breaks; '→' transition, '+' assembly, ';' disassembly, and 'ϕ_2' temperature 2), where A-H represent component information.

I fits$_{360}$ J → I+J	ϕ_2 breaks I+J → I ; J
K fits$_{360}$ L → K+L	ϕ_2 breaks K+L → K ; L
M fits$_{180}$ N → M+N	ϕ_2 breaks M+N → M ; N
O fits$_{90}$ P → O+P	ϕ_2 breaks O+P → O ; P
Q fits$_{90}$ R → Q+R	ϕ_2 breaks Q+R → Q ; R
S fits$_{90}$ T → S+T	ϕ_2 breaks S+T → S ; T

Table 4.2: 3D interaction rules (fits and breaks; '→' transition, '+' assembly, ';' disassembly, and 'ϕ_2' temperature 2), where I-T represent component information and 360, 180, and 90 represent component rotational interactions.

Seed Components	not required	required
Parallel Self-Assembly	yes	no
Number of Tile Types at an Assembly Location	multiple	one
2D/3D	2DcTAM/3DcTAM	2D
Rotations	2D/3D	none
One-Pot-Mixture	yes	yes

Table 4.3: Model features of the cTAM compared to the aTAM

(Bhalla et al. 2007). This model was later replaced with a more computationally efficient tile-based model (Bhalla et al. 2010; Bhalla & Bentley in print; Bhalla et al. 2011a), which is better suited for comparison to other tile-based models of self-assembly. The tile-based model used here is an extension to the aTAM and combines features of Lego World, and is referred to as the 2D concurrent Tile Assembly Model and the 3D concurrent Tile Assembly Model (2DcTAM and 3DcTAM; distinguished spatially using 2D and 3D components respectively). The 2DcTAM and 3DcTAM are better suited to the type of self-assembling systems used in this work, by accommodating for component rotation and allowing for concurrent self-assembly. The features of the aTAM are compared to the features of the 2DcTAM and 3DcTAM in Table 4.2.

The 2DcTAM and the 3DcTAM follow a common algorithm. Pseudocode for the *concurrent Tile Assembly Algorithm* is provided on the next page. Figure 4.3 (page 90) provides an example of the assembly process in the 2DcTAM. A post-evaluation of the set of resulting self-assembled structures at the conclusion of the algorithm is sufficient in this work, as the set of resulting self-assembled structures is of more concern than environmental constraints in this work. Since assembly is considered concurrently and more than one tile (component) type can be used at an assembly location, errors must be accounted for and include: *neutral site*, *uncomplimentary information*, and *boundary* violations, as well as the requirement of an *assembly path* (Figure 4.4, page 91).

Algorithm *concurrent Tile Assembly Algorithm*

Input: a multiset of tiles FounderSet

Output: set of self-assembled target structures and remaining tiles

```
while there are tiles in FounderSet with open assembly locations
  do select a random R tile or substructure from FounderSet that has a tile with an open assembly location L
     remove R from FounderSet
     set L to unmatchable select all tiles and substructures from FounderSet that have tiles with open complementary information to L and place in a list AssemblyCandidateList
     while L is unmatchable AND the size of AssemblyCandidateList $> 0$
       do select at random a tile or substructure T from AssemblyCandidateList
          if R can be added to T (without incurring an assembly violation, all applicable assembly locations, including their rotational information, must not conflict when adding two substructures together)
            then add R to T
                 set L and all applicable open assembly locations in R and T to match
                 remove R from FounderSet
                 add T to FounderSet
            else remove T from AssemblyCandidateList
return FounderSet
```

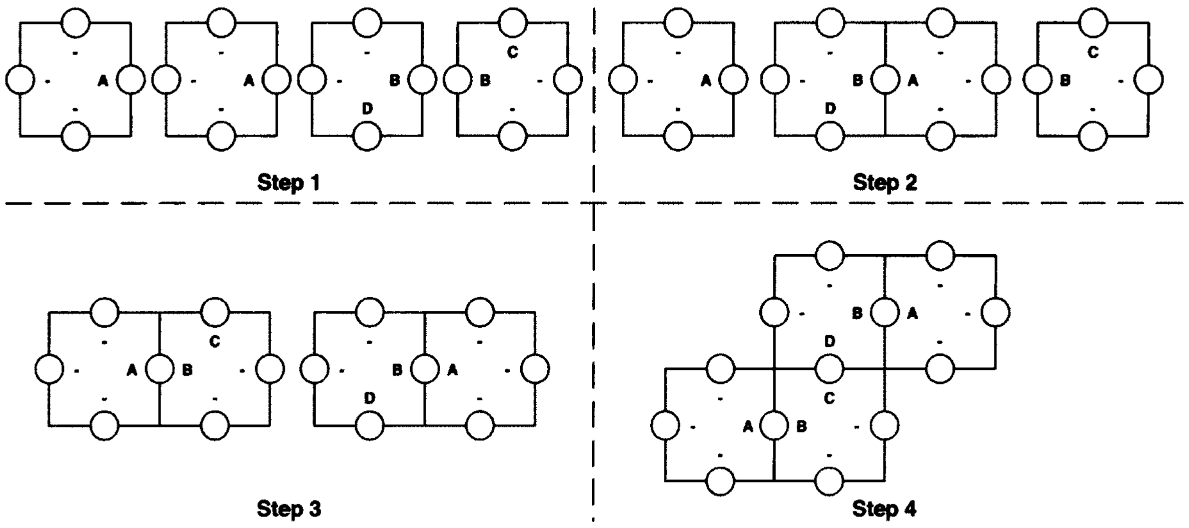


Figure 4.3: 2DcTAM example steps.

4.3 Level Three: Physical Realization of Rule Set

While in levels one and two it is assumed component and environment information can be perfectly achieved, in level three this theoretical information must be physically instantiated, often resulting in precise choices of materials, energy, and forms for components and their environment. The physical systems used in this work have similar characteristics. The goal of each system is to investigate whether or not a set of mechanical, macroscale components (with concave/convex polygon or polyhedron shapes) can self-assemble into a target structure (with a polygon perimeter or polyhedron volume form, with symmetric/asymmetric features). Components are either confined to a surface (2D) or suspended in a fluid (3D). Environment vibrations transfer energy to the components, causing components to move around and interact with one another. These two new automated environments improve upon hand-controlled environments (Bhalla & Bentley 2006; Olson et al. 2007; Tibbits 2010; Hosokawa et al. 1996) and are similar to Miyashita et al. (2009), using a tray surface for 2D systems (but not using of a fluid in the tray) and extended for 3D systems.

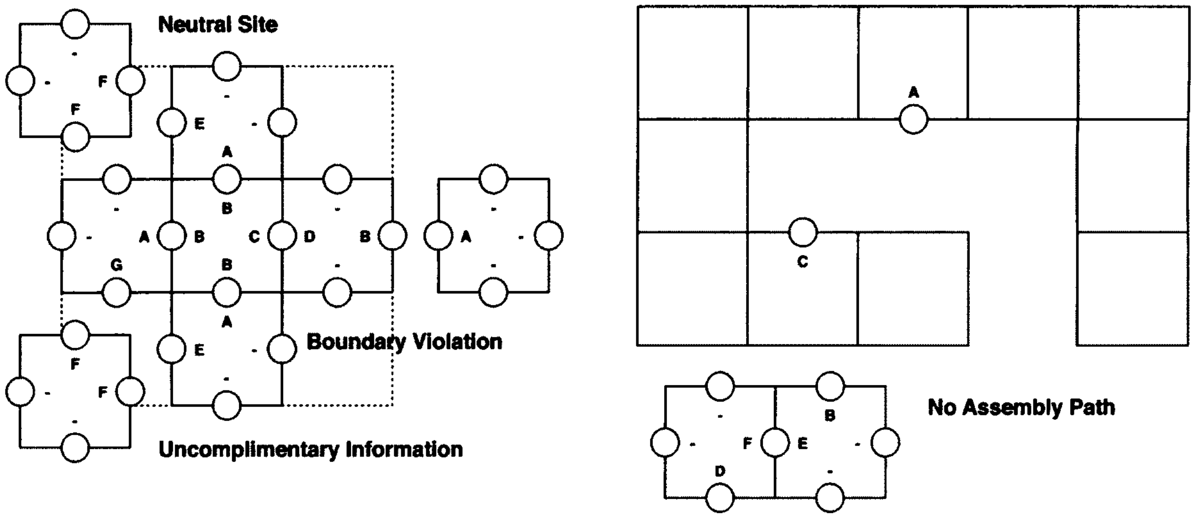


Figure 4.4: Examples of assembly violations (left) and no assembly path (right).

Permanent magnets embedded in the components are used to enable components to attract and repel one another. The approach to physically encoding information in terms of a component design space is inspired by chemistry, particularly molecular interactions. The components presented here use a simplified set of shape primitives and magnetic patterns (using multiple permanent magnets). The set of shape primitives use key and lock shapes (Bhalla & Bentley 2006; Terfort et al. 1997) and a neutral shape to allow for the creation of closed target structures (Bhalla & Bentley 2006). By using magnetic patterns, these new components are conceptually similar to DNA tiles (Winfree et al. 1998a), with the distinction that 3D DNA tiles have not been physically created at the time of writing. Magnetic patterns along the edges of 2D components or on the faces of 3D components (Garcias et al. 2002), in addition to key and lock shapes, enable component-to-component interactions. The role of component shape and magnetic interactions are contrasted to demonstrate how to leverage particular aspects of component interactions during self-assembly.

4.3.1 Component Design Space

Fisher's key-lock principle (*section 2.2.1: Physical Information Encoding in Nature*) is extended in this work to a key-lock-neutral concept (neutral meaning where no bonding is possible) and apply it to self-assembly. This concept is central to the component design space used here. The design space defines the set of physically feasible designs. Here, the design space is a combination of a shape space and an assembly protocol space. The shape space is based on the key-lock-neutral concept to define component shape. The assembly protocol space uses magnetism to define bonding. Together, they can be used to enable the self-assembly process.

However, the shape space and the assembly protocol space must be used in such a way to give the component design space two properties: (1) using shapes to create stable joints between complementary components, and (2) creating the ability for components to selectively bond to corresponding components and not to conflicting components (Bhalla & Bentley 2006). The first property is achieved by using both concave and convex shapes, creating joints which are less likely to disassemble during collisions with other components and the environment. The second property is achieved by placing the magnets in the interior of a non-magnetic material. By not allowing components to join directly together, components have a higher degree of freedom to move around and interact with one another. This is an important aspect to the definition of self-assembly, requiring adaptive component interactions (*section 1.2: Thesis Hypothesis*).

A continuous version of this design space for self-assembly was first used in (Bhalla 2004; Bhalla & Bentley 2006; Kaewkamnerdpong et al. 2007). In this work, the design space is a discrete version. Two implementations in both 2D and 3D are contrasted to demonstrate how the design space can be leveraged to reduce potential component-to-component interaction errors from occurring during the self-assembly process.

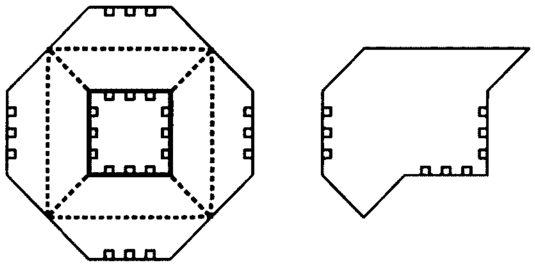


Figure 4.5: Design space for the first 2D physical information encoding scheme (left), and an example component showing the shortcoming of neighbouring lock shapes (right).

4.3.2 2D Physical Information Encoding Schemes

Different materials and manufacturing techniques to construct 2D components are used to demonstrate how they can be used in two 2D physical information encoding schemes to reduce potential errors from occurring during the self-assembly process.

First 2D Physical Information Encoding Scheme

This 2D physical information encoding scheme (Figure 4.5) was used in the original implementation of the three-level approach (Bhalla et al. 2007). In this case, components and the environment were hand-built from foam board, and the environment was shaken in 2D. A component's base shape is a square, where a shape primitive is applied to each edge: key (convex), lock (concave), and neutral (linear). Three magnets are associated with keys and locks, but not with neutral sites. Magnets are placed within the sides of the components, and covered with foam, to allow for selective bonding. A single magnet is used at each location in the 3-magnetic-bit pattern (magnetic south/north arbitrarily assigned to one and zero). By manipulating the designation of the 3-magnetic-bit patterns to keys and locks, convex key-to-key errors can be reduced (Table 4.4). However, this is at the expense of not being able to minimize key-to-lock errors. Due to the challenges of building components from foam by hand, the shortcoming of this shape space was that neighbouring lock shapes on the same component are not well defined (Figure 4.5).

Lock	000	A	A fits B → A+B	ϕ_2 breaks A+B → A ; B
Lock	001	C	C fits D → C+D	ϕ_2 breaks C+D → C ; D
Lock	100	E	E fits F → E+F	ϕ_2 breaks E+F → E ; F
Lock	010	G	G fits H → G+H	ϕ_2 breaks G+H → G ; H
Key	111	B	B fits A → B+A	ϕ_2 breaks B+A → B ; A
Key	110	D	D fits C → D+C	ϕ_2 breaks D+C → D ; C
Key	011	F	F fits E → F+E	ϕ_2 breaks F+E → F ; E
Key	101	H	H fits G → H+G	ϕ_2 breaks H+G → H ; G

Table 4.4: First key and lock designations to the 3-magnetic-bit patterns (red/zero and blue/one represent magnetic south and north respectively)

The corresponding environment information for this first 2D design space matched the environment for the continuous design space: materials (foam board), shaking motion (by hand randomly in 2D, parallel to the surface of the tray), and resulting component-to-component and component-to-environment interactions (for corresponding and conflicting components). Five physical self-assembling systems were created, Figure 4.6, by building components using this first 2D design space (and varying the number and types of components) and their corresponding tray environment. Despite the shortcoming of this shape space, the five systems were able to achieve their target structures.

Second 2D Physical Information Encoding Scheme

A solution to the shortcoming of the first 2D physical information encoding scheme is achieved by using rapid prototyping to fabricate components (Figure 4.7, page 96). As with the first 2D shape space, three shape primitives are used in association with a base

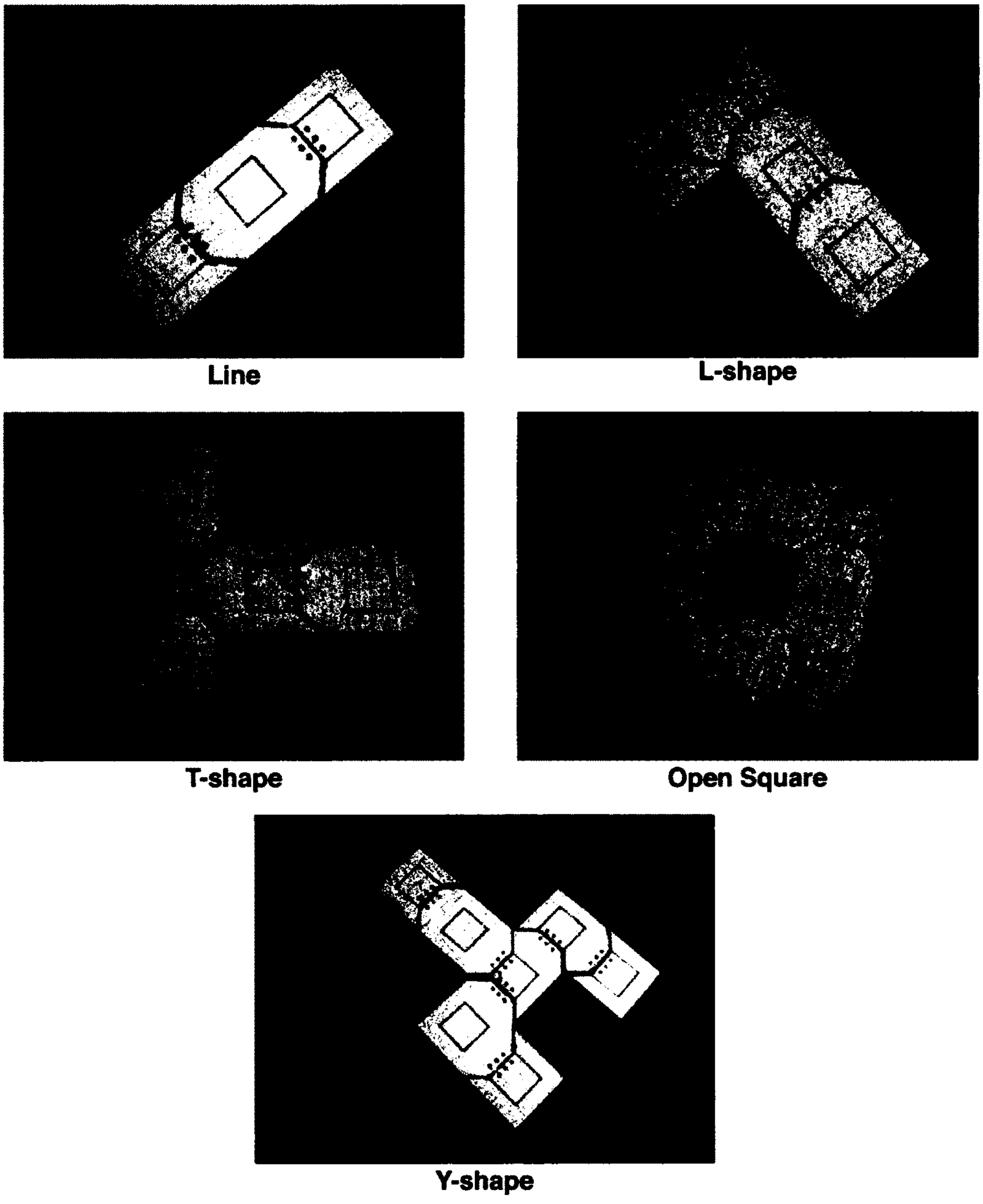


Figure 4.6: Self-assembling systems using the first 2D physical information encoding scheme (Bhalla et al. 2007).

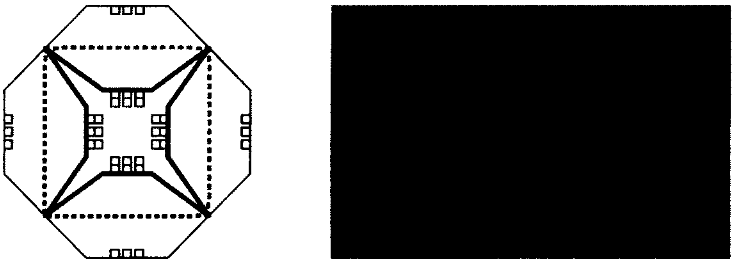
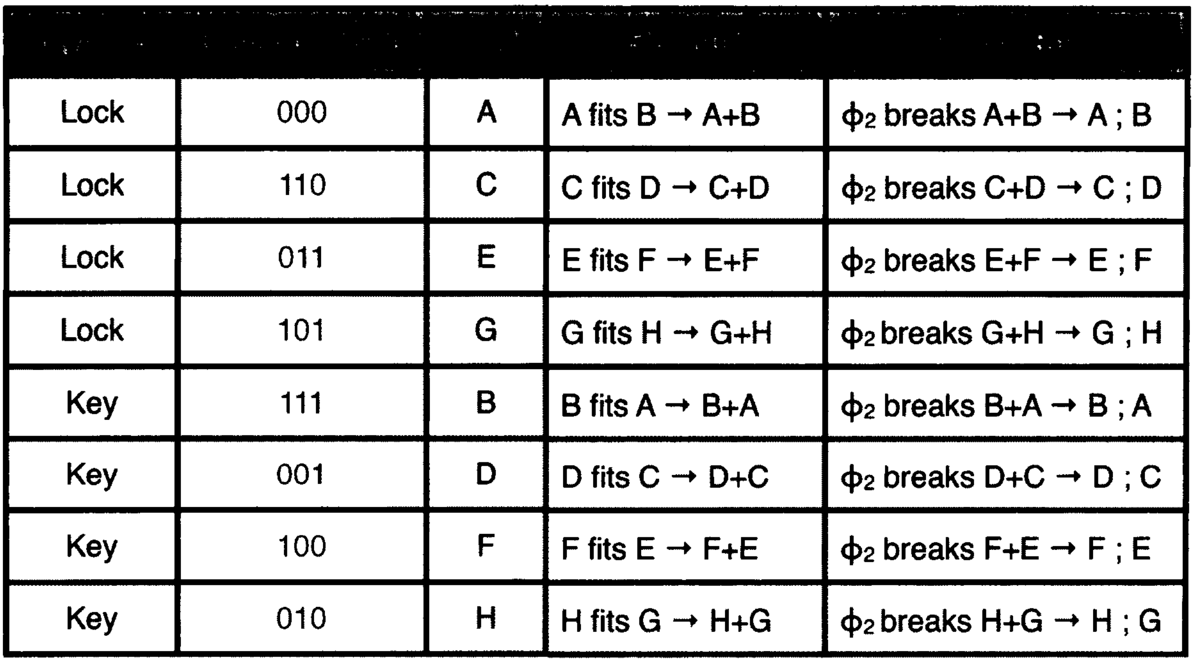


Figure 4.7: Design space for the second 2D physical information encoding scheme (left), and an example component with information markers.

square shape. Smaller and stronger components can be built, since rapid prototyping is used. As a result, neighbouring lock shapes on the same component are well defined. The main body of a component is made from plastic. Similar to the first 2D assembly protocol space, a 3-magnetic-bit pattern is used. Due to greater precision in fabricating components using rapid prototyping, lock-to-lock interactions never occur and more magnets can be placed within the sides of components (using an air gap along an edge for selective bonding). Instead, the 3-magnetic-bit pattern uses one magnet in each position associated with a key and two magnets in each position associated with a lock (Table 4.5). This ensures strong bonding between keys and locks, and weak bonding between keys. Weak bonding can be avoided through an appropriate physical environment temperature. Therefore, key-to-key matching errors can be avoided and key-to-lock matching errors can be reduced through proper designation of the 3-magnetic-bit patterns to keys and locks.

Information markers on the top surfaces of components, using red and blue paint, are used to identify the 3-magnetic-bit pattern used with a key or lock (Figure 4.7). A 2D component's base shape is 10 mm^2, and can have a maximum width or length of 20 mm^2. Specifications are detailed in *Appendix A.2: 2D Components.*

The corresponding environment information for this second 2D design space contrasts the environment for the first 2D design space: in materials (fabricated from plastic using



Lock	000	A	A fits B → A+B	ϕ_2 breaks A+B → A ; B
Lock	110	C	C fits D → C+D	ϕ_2 breaks C+D → C ; D
Lock	011	E	E fits F → E+F	ϕ_2 breaks E+F → E ; F
Lock	101	G	G fits H → G+H	ϕ_2 breaks G+H → G ; H
Key	111	B	B fits A → B+A	ϕ_2 breaks B+A → B ; A
Key	001	D	D fits C → D+C	ϕ_2 breaks D+C → D ; C
Key	100	F	F fits E → F+E	ϕ_2 breaks F+E → F ; E
Key	010	H	H fits G → H+G	ϕ_2 breaks H+G → H ; G

Table 4.5: Second key and lock designations to the 3-magnetic-bit patterns (red/zero and blue/one represent magnetic south and north respectively)

rapid prototyping) and shaking motion (in a 2D orbital motion, automated using a Maxi Mix II Vortex Mixer). Achieving an appropriate environment temperature, to maintain bonds between complementary components and break bonds between conflicting components, was extremely difficult. As with the two previous physical self-assembling systems examples, the environment temperature corresponds to a mechanical shaking level. This mechanical shaking level is a combination of the speed of the mixer, the total number of components in the system, and the size of the tray environment. Also, the Maxi Mix II Vortex Mixer is not purely an orbital shaker. A spring joint in the mixer adds variability to the shaking motion. This variability is another factor in the mechanical shaking level. However, a rpm setting of 1,050 was found to create the appropriate temperature (abstractly equivalent to one). Specifications are detailed in *Appendix B.2: 2D Environment*.

4.3.3 3D Physical Information Encoding Schemes

With the benefits of rapid prototyping being demonstrated in fabricating 2D components, rapid prototyping is also used to fabricate 3D components. This form of precision manufacturing allows for the manipulation of magnetic patterns which are used in two 3D physical information encoding schemes to reduce rotation errors during component-to-component interactions during the self-assembly process.

First 3D Physical Information Encoding Scheme

A 5-magnetic-bit pattern defines the assembly protocol space (Figure 4.8). The five permanent magnetic discs can be added to a face of a cube-based component. Of the thirty-two total 5-magnetic-bit patterns, there are six unique complementary pairs of patterns when considering planar rotation of a component's face (Figure 4.8). These six codes encapsulate rotational information for component-to-component interactions, where two pairs encapsulate 360°, one pair encapsulates 180°, and three pairs encapsulate 90° rotational component-to-component interactions.

A key-lock-neutral concept defines the shape space (Figure 4.9). Either a key, lock, or neutral shape is used on each face of a component. A pair of complementary key and lock shapes is used to create stable joints between complementary components. A 5-magnetic-bit pattern is placed within a key or a lock. However, the magnets are not flush with the surface, creating an air gap, to allow for selective bonding between components by maintaining component-to-component interactions to be adjustable (*section 1.2: Thesis Hypothesis*). In this context, neutral means where no assembly is possible. To simplify the evaluation of correct self-assembly bond formations at the conclusion of the self-assembly process (visually by the user), coloured circles are used (when possible) on neutral sites to identify neighbouring 5-magnetic-bit patterns (referred to as interaction markers). A 3D component's base shape is 15 mm^3, and can have a maximum height of

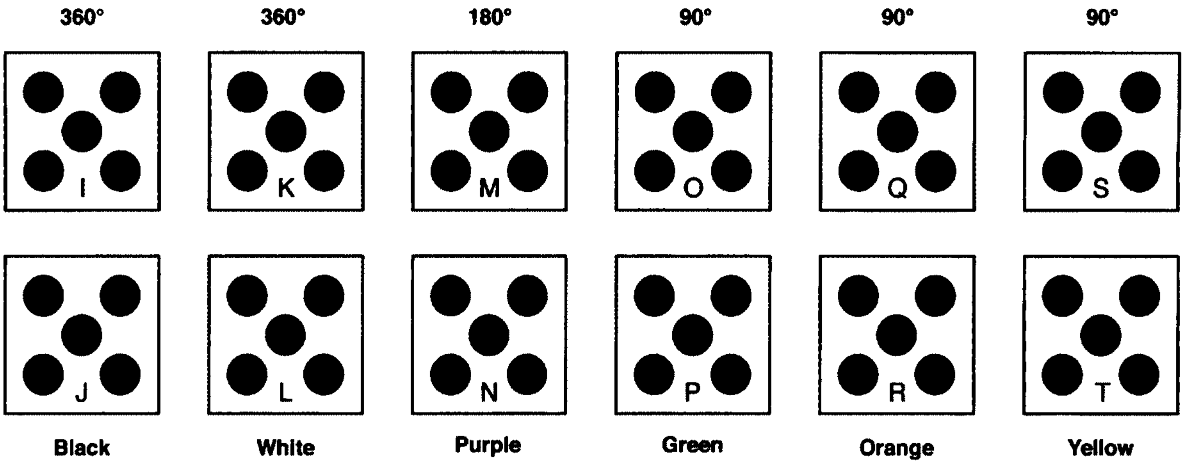


Figure 4.8: Six pairs of unique 5-magnetic-bit patterns, where blue and red represent magnetic north and south respectively, along with rotational properties (360°,180°, and 90°), and colours associated with each pair (used as paint markers on components to indicate information associated with component-to-component interactions).

20 mm^3. Specifications are detailed in *Appendix A.3: 3D Components.*

A linear representation of the magnets on a component's face are listed as: centre, bottom-right, bottom-left, top-left, and top-right. Again, one represents magnetic north and zero represents magnetic south arbitrarily. In this first 3D physical information encoding scheme, a single permanent magnet is used in each location of the 5-magnetic-bit patterns. Using this magnetic condition, the 5-magnetic-bit patterns can be assigned to keys and locks to reduce key-to-key error interactions (Table 4.6). In this example,

Figure 4.9: 3D component shape space (left; showing the base component with information markers, the minimum component with lock shapes on all six faces, and the maximum component with key shapes on all six faces), and example components (right).

Lock	00000	I	I fits$_{360}$ I → I+J	ϕ_2 breaks I+J → I ; J
Lock	10000	K	K fits$_{360}$ L → K+L	ϕ_2 breaks K+L → K ; L
Lock	01010	M	M fits$_{180}$ N → M+N	ϕ_2 breaks M+N → M ; N
Lock	01100	O	O fits$_{90}$ P → O+P	ϕ_2 breaks O+P → O ; P
Lock	11000	Q	Q fits$_{90}$ R → Q+R	ϕ_2 breaks Q+R → Q ; R
Lock	01000	S	S fits$_{90}$ T → S+T	ϕ_2 breaks S+T → S ; T
Key	11111	J	J fits$_{360}$ I → I+J	ϕ_2 breaks J+I → J ; I
Key	01111	L	L fits$_{360}$ K → L+K	ϕ_2 breaks L+K → L ; K
Key	10101	N	N fits$_{180}$ M → N+M	ϕ_2 breaks N+M → N ; M
Key	10011	P	P fits$_{90}$ O → P+O	ϕ_2 breaks P+O → P ; O
Key	00111	R	R fits$_{90}$ Q → R+Q	ϕ_2 breaks R+Q → R ; Q
Key	10111	T	T fits$_{90}$ S → T+S	ϕ_2 breaks T+S → T ; S

Table 4.6: First key and lock designations to the 5-magnetic-bit patterns (red/zero and blue/one represent magnetic south and north respectively)

the worst possible match between magnets that can occur is a three out of five match (e.g. I interacting with N).

Second 3D Physical Information Encoding Scheme

In this second example, a single permanent magnet is used in each location of the 5-magnetic-bit patterns which are assigned to keys, and two permanent magnets are used in each location of the 5-magnetic-bit patterns which are assigned to locks. This magnetic condition is used to take advantage of lock-to-lock interactions not being possible, and allows for the reduction of key-to-lock error interactions (Table 4.7). Although key-to-

Lock	00000	I	I fits$_{360}$ I → I+J	ϕ_2 breaks I+J → I ; J
Lock	10000	K	K fits$_{360}$ L → K+L	ϕ_2 breaks K+L → K ; L
Lock	01010	M	M fits$_{180}$ N → M+N	ϕ_2 breaks M+N → M ; N
Lock	10011	P	P fits$_{90}$ O → P+O	ϕ_2 breaks P+O → P ; O
Lock	00111	R	R fits$_{90}$ Q → R+Q	ϕ_2 breaks R+Q → R ; Q
Lock	10111	T	T fits$_{90}$ S → T+S	ϕ_2 breaks T+S → T ; S
Key	11111	J	J fits$_{360}$ I → I+J	ϕ_2 breaks J+I → J ; I
Key	01111	L	L fits$_{360}$ K → L+K	ϕ_2 breaks L+K → L ; K
Key	10101	N	N fits$_{180}$ M → N+M	ϕ_2 breaks N+M → N ; M
Key	01100	O	O fits$_{90}$ P → O+P	ϕ_2 breaks O+P → O ; P
Key	11000	Q	Q fits$_{90}$ R → Q+R	ϕ_2 breaks Q+R → Q ; R
Key	01000	S	S fits$_{90}$ T → S+T	ϕ_2 breaks S+T → S ; T

Table 4.7: Second key and lock designations to the 5-magnetic-bit patterns (red/zero and blue/one represent magnetic south and north respectively)

key interactions will still occur, the environment temperature can be adjusted to break magnetically weak key-to-key bonds and maintain magnetically strong key-to-lock bonds. Furthermore, there is more than one configuration of 5-magnetic-bit patterns assigned to keys and locks to reduce key-to-lock error interactions.

The corresponding physical environment information to these two 3D component physical encoding schemes consists of a jar to provide a boundary for components, the placement of the jar in a rack on an orbital shaker to provide energy to the system (a parameter in setting environment temperature), and mineral oil in the jar to allow components to freely move in three spatial dimensions. Mineral oil is used to prevent the permanent magnets from corroding, and to provide the appropriate viscosity. The design of the environment was discovered through extensive preliminary trials and experiments. Specifications are detailed in *Appendix B.3: 3D Environment*.

4.4 Summary

This chapter introduced the three-level approach for designing self-assembling systems via physically encoded information. Three levels of this design methodology include: (1) definition of rule set, (2) virtual execution of rule set, and (3) physical realization of rule set. The objective of of this approach is to provide an inherently bottom-up manner to create components and/or their environment for addressing the scale problem of designing self-assembling systems. This is achieved by being able to directly map a set of rules to physical systems.

At level one, component, environment, and system rules were presented. These rules provide a high-level, abstract description of the type of self-assembling systems used in the preceding experiments. As well, 3D component interaction rules with rotational information were presented. These rules incorporate component rotations in temperature one systems.

At level two, a computationally efficient tile-based model, the 2DcTAM and the 3DcTAM, was discussed. These two tile-based models are distinguished from the aTAM and its extensions, because they consider concurrent self-assembly and incorporate component rotations. As a result, assembly violations need to be considered in the two models. The 2DcTAM and the 3DcTAM are both Turing universal at temperature one, just as the rgTAM (Patitz et al. 2011) and 3D aTAM (Cook et al. 2011) have been proven to be Turing universal at temperature one (*section 2.2.5: The Complexity of Self-Assembly*).

At level three, a component design space that consists of a shape space and an assembly protocol space was described. A key-lock-neutral concept defines the shape space. 3-magnetic-bit and 5-magnetic-bit patterns defines the assembly protocol space. Furthermore, this design space was described in the context of two physical information encoding schemes, for both 2D and 3D systems. It was shown that by varying the number of magnets in each location of the magnetic-bit patterns, component-to-component interaction errors can be reduced. Also, the 5-magnetic-bit patterns encode 360°,180°, and 90° 3D component rotational information. Lastly, the advantages of rapid prototyping to fabricate components were presented, particularly in precision manufacturing of the shape space. Along with fabricated components, automated environments were also presented, to create reproducible self-assembling systems. The second 2D and 3D physical encodings schemes are used in a series of experiments in *Chapter 5: Programming Self-Assembling Systems*, *Chapter 6: Evolving Self-Assembly Rule Sets*, and *Chapter 7: Staging the Self-Assembly Process.*

Chapter 5

Programming Self-Assembling Systems

"The connection between self-assembly and computation suggests that a shape can be considered the output of a self-assembly "program", a set of tiles that fit together to create a shape" (Solveichik & Winfree 2007). This quote summarizes the intent of the first set of experiments, which was to test the feasibility of the three-level approach to program (design) self-assembling systems. In these programming experiments, the three-level approach is used directly to demonstrate a programming paradigm where rules are hand-designed to test the second 2D and 3D physical information encoding schemes, Table 4.5 (page 97) and Table 4.7 (page 101) respectively. Research pertaining to the 2D programming experiments is published in Bhalla & Bentley (in print), and the 3D programming experiments in Bhalla et al. (2011a).

5.1 Hypothesis Statement

The hypothesis for the programming experiments was,

Hypothesis: physical information can be used to enable components to self-assemble into closed target structures with symmetric/asymmetric features.

The three-level approach, by first defining sets of rules (level one), was used to test this hypothesis, virtually (level two) and physically (level three). Three 2D programming (2DP) experiments and five 3D programming (3DP) experiments (referred to as 2DPE1 to 2DPE3 and 3DPE1 to 3DPE5 respectively) were used to test this hypothesis. A target

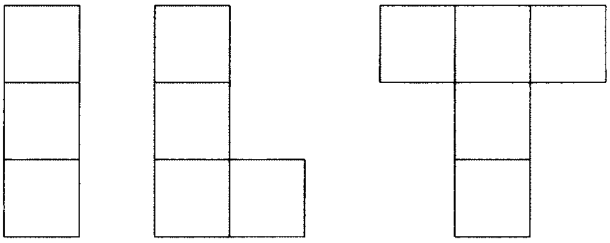


Figure 5.1: From left to right: target structures for the 2DP experiments corresponding to 2DPE1 to 2DPE3 respectively.

structure was assigned to each experiment (provided in sections *5.2: 2D Programming Experiments and Results* and *5.3: 3D Programming Experiments and Results*). Enough components were supplied to create up to three 2D and 3D target structures. The number of target structures corresponds to the capacity of the environments used. Ten trials were run for each experiment[1]. A trial was evaluated to be successful if all three 2D and 3D target structures were created at level two, and if at least one 2D and 3D target structure was created at level three. The independent variable in these experiments was the set of components, defined by their type and frequency. The dependent variable is the resulting self-assembled structures. For each experiment, a *programmed* component set was specified along with a *random* component set to test the independent variable.

5.2 2D Programming Experiments and Results

Three 2D programming experiments were conducted, where a target structure was assigned to each experiment (Figure 5.1). The experimental procedures and results are described according to the three-level approach: defining self-assembly rule sets (level one), modelling using 2DcTAM (level two), and physically testing systems using the second 2D physical information encoding scheme (level three).

[1] An additional five trials for each 2D experiment, to the original five trials presented in (Bhalla & Bentley in print), were conducted to collect further data.

These three target structures were chosen because they offer various degrees of complexity in terms of the number of components required and symmetric/asymmetric features in their morphology. Consequently, each target structure cannot be created by pattern formation exclusively. Therefore, it was appropriate to determine if the components had sufficient information to achieve the target structures by self-assembly.

5.2.1 Level One: Rule Set for 2DP Experiments

At level one, component rule sets were specified for the level two and level three trials. Programmed component sets were created using a combination of top-down decomposition and bottom-up trial and error. The morphology of a single target structure was used to identify the *connectivity* (number and position of neighbouring components) of each component, and determine the number of unique components (based on connectivity) and their frequency. The connectivity of each unique component established the symmetric and asymmetric information required, and was used to assign component information to create each component type. A detailed example of designing a self-assembling system is provided in *section 8.2: Leveraging Limited Rule Sets.* After this initial set was specified, the quantity of each component type was multiplied by three to create the maximum number of target structures for each experiment. The difficulty was in verifying if a correct set of components were specified when considering parallel construction without the use of seed components, and the self-assembly process not resulting in errors.

For the random component sets, component types were specified by selecting with uniform probability the information (A to H, and –) assigned to each information location. Then the quantity of each component type was multiplied by three to create the maximum number of target structures for each experiment, conducted at level two and three. Component rules and the quantity of each component are provided in Table 5.1. The system rules from Table 4.5 (page 97) were applicable to both groups.

2DPE1	P	6 X (B,-,-,-), 3 X (A,-,A,-)
	R	6 X (F,B,-,D), 3 X (D,D,D,H)
2DPE2	P	6 X (B,-,-,-), 3 X (-,D,-,A), 3 X (A,-,C,-)
	R	6 X (B,B,C,F), 3 X (-,A,A,C), 3 X (C,C,D,D)
2DPE3	P	9 X (B,-,-,-), 3 X (-,A,G,A), 3 X (H,-,A,-)
	R	9 X (-,E,D,D), 3 X (E,B,G,E), 3 X (G,B,B,B)

Table 5.1: Programmed and random (P/R) component sets (represented as '# × (Top, Left, Bottom, Right)', where # represents quantity and the directions refer to component information) for the 2DP experiments (EX)

5.2.2 Level Two: Virtual Execution of Rule Set for 2DP Experiments

For each 2DP experiment, the 2DcTAM (*section 4.2: Level Two: Virtual Execution of Rule Set*) was used to virtually evaluate the ability of each rule set (programmed or random) to create its respective target structure.

Level Two: Experimental Setup for 2DP Experiments

Component rules from Table 5.1 and the environment rules were mapped to an abstract representation appropriate for the 2DcTAM. Each component's shape was a unit square. The environment rule associated with boundary size was set to 10×10 units (a representation of width×depth, and the ratio between component and environment size). The temperature parameter was set to one. A different random seed was used to initialize the 2DcTAM for each trial.

Level Two: Experimental Results for 2DP Experiments

At level two, the 2DcTAM was used to evaluate each trial. Each of the programmed component sets successfully created three of their applicable target structures. These results show that even with no component acting as a seed, and permitting component rotation, it is still possible to successfully create target structures. Furthermore, these results also show that it is possible to create multiples of the same target structure concurrently, when appropriate component information is used.

In contrast, none of the random component sets successfully created at least one target structure. There are two reasons for the unsuccessful random component sets. The first reason, which applies to the random component sets used in 2DPE1 and 2DPE3, was that no compatible fits rule existed for the information encoded in the components. As a result, no assembly could take place with these components. The second reason, which applies to the random component set used in 2DPE2, is that multiple fits rules apply to either multiple component types or to multiple information locations within the same component type. There are two information locations labelled D in the third component type. These two information locations can bond to information locations labelled C in the first and second component type. In addition, since multiples of the third component type exist, they can also bond to each other. As a result, the generated structures are not stable, meaning the same structures would not emerge from different executions of the 2DcTAM.

Fisher's Exact Test (one-sided) for analyzing binary data (Cox & Snell 1989), was used to analyze the results of the level two 2DP experiments. The results are statistically significant, with a p-value of 0 (Table 5.2). These successful results support the programming experiment hypothesis at level two in the context of the 2DP experiments.

5.2.3 Level Three: Physical Realization of Rule Set for 2DP Experiments

With the success of each system using a programmed component set, a level three translation was performed for each 2DP experiment. A level three translation was not performed on the systems using a random component set, since they were not successful.

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2DPE1	Programmed	10	0	0
	Random	0	10	
2DPE2	Programmed	10	0	0
	Random	0	10	
2DPE3	Programmed	10	0	0
	Random	0	10	

Table 5.2: The number of successful and unsuccessful trials for each programmed and random 2DP experiment (2DPE1 - 2DPE3) at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data

Level Three: Experimental Setup for 2DP Experiments

Component mapping followed Table 4.5 (page 97). For each trial, components were randomly placed on the surface of the tray, which was mounted on an orbital shaker (their environment). Each system was shaken for 20 minutes. The state a system was recorded at the conclusion of each trial, observations included: the *number of target structures* created, the number of *matching errors* (between conflicting physical information, where no fits rule is applicable), and the number of *assembly errors* (partial attachment where a fits rule is applicable). Details regarding the physical experiment setup for the 2DP experiments is provided in *section C.2: Experimental Procedure for the 2D Systems.*

Level Three: Experimental Results for 2DP Experiments

Figure 5.2 shows the number of target structures created in the 2DP experiments. Figure 5.3 (page 111) shows an example of a successful trial for each experiment. One reason for unsuccessful trials in 2DPE3, was the resulting spatial relationship between emerging substructures. In 2DPE3 trial 2, the components self-assembled into two substructures

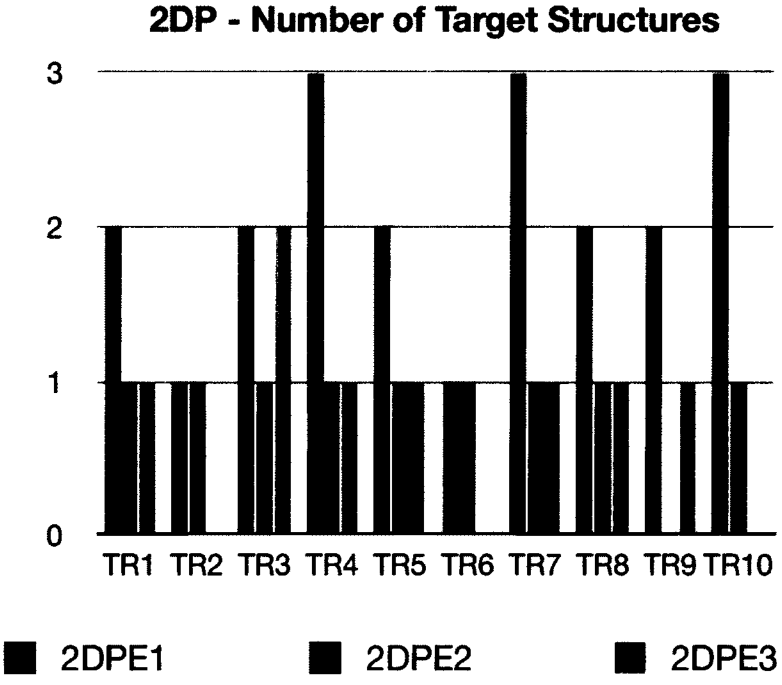
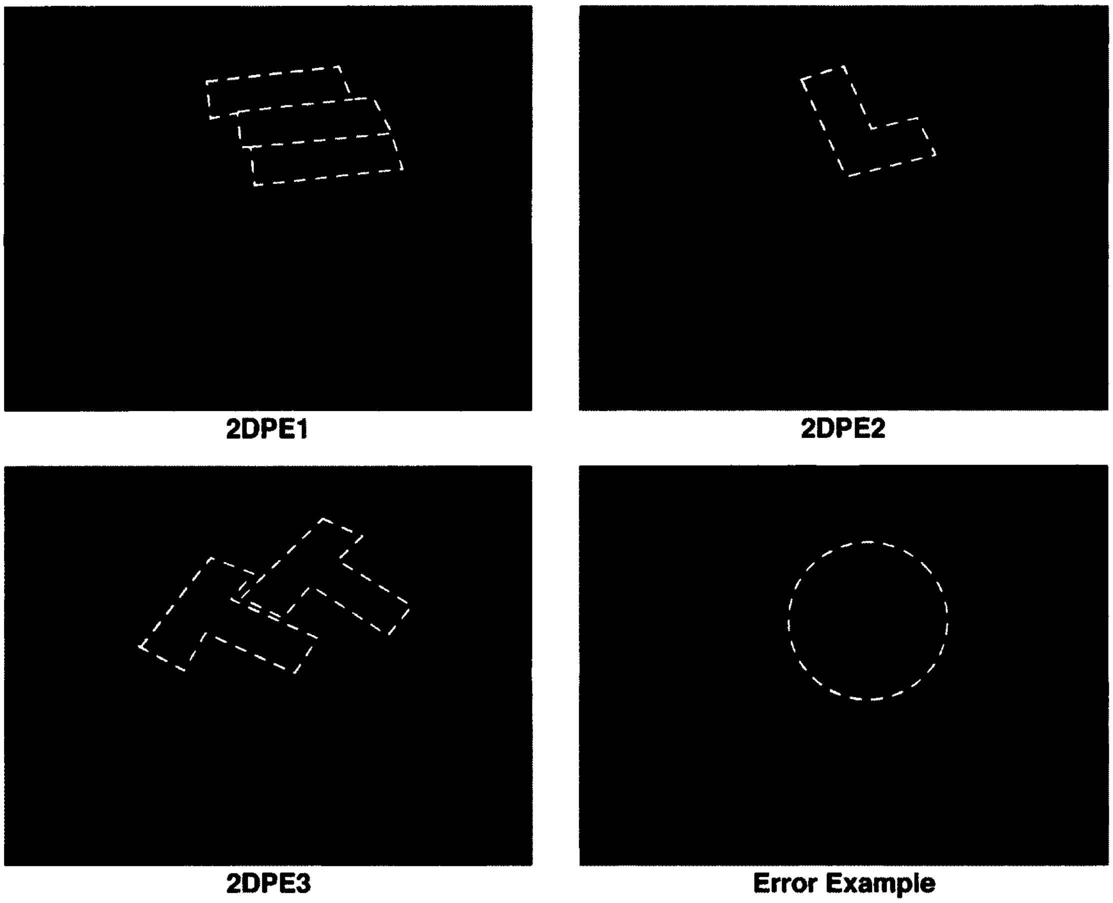


Figure 5.2: Number of target structures created at the end of each trial (TR1 - TR10), for each 2DP experiment (2DPE1 - 2DP3).

that would not allow the self-assembly process to continue (Figure 5.3).

No assembly errors were present at the end of each trial for each of the three 2DP experiments. Only 2DPE1 had zero matching errors, due to the single physical code pair used. 2DPE2 had more matching errors in comparison to 2DPE3 (Figure 5.4, page 112), which can be explained by two reasons. The first reason is that the two physical code pairs used in 2DPE2 have the highest matching error possibility. The second reason is that the total number of components used in 2DPE2 are fewer, and as a result, there are less components moving around to break apart assembly errors.

It was qualitatively observed that self-repair was possible in the systems. For example, if two complementary components formed an assembly error, typically a collision with the environment (the side of the tray wall) would result in a proper assembly. In practice it was a challenge to implement an environment temperature of one.



2DPE1

2DPE2

2DPE3

Error Example

Figure 5.3: Photographs of successful 2DP experiments (2DPE1 - 2DPE3) and an example error (trial from 2DPE3), where adjacent components to the outlined shapes are not attached to the self-assembled structures (resting positions when energy was removed from the environment).

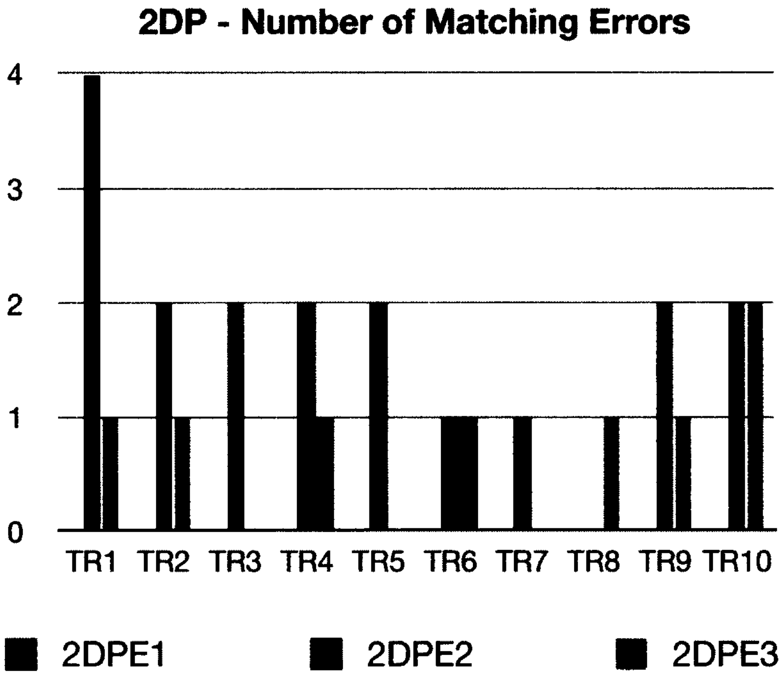


Figure 5.4: Number of matching errors at the end of each trial (TR1 - TR10), for each 2DP experiment (2DPE1 - 2DP3).

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2DPE1	Programmed	10	0	0
2DPE2	Programmed	9	1	0
2DPE3	Programmed	7	3	0.002

Table 5.3: The number of successful and unsuccessful trials for the 2DP experiments at level three, with corresponding p-values (calculated using Fisher's Exact Test (one-sided) for analyzing binary data with respect to 0 successful and 10 unsuccessful trials for the random component sets for each 2DP experiment from their level two results)

The 2DP experiments at level three were analyzed using Fisher's Exact Test (one-sided) for analyzing binary data (Cox & Snell 1989). All three 2DP experiments are statistically significant at the 0.01 level (meaning there is a 99% chance the results are not due to chance, Table 5.3). The successful 2DP experiments provide evidence to support the hypothesis that physical information can be used to enable a set of components to self-assembly into 2D closed target structures with symmetric/asymmetric features.

5.3 3D Programming Experiments and Results

Five 3DP experiments were conducted. Figure 5.5 provides the corresponding target structures for the five 3DP experiments. The experimental procedures and results are described according to the three-level approach: defining a self-assembly rule set (level one), modelling using 3DcTAM (level two), and physically testing systems using the second 3D physical information encoding scheme (level three).

These five target structures were chosen since they offer degrees of complexity in terms of the number of components and their concentration and symmetric/asymmetric features in the target structures. Consequently, the five target structures cannot be created by using the underlying cubic lattice pattern of component formations exclusively.

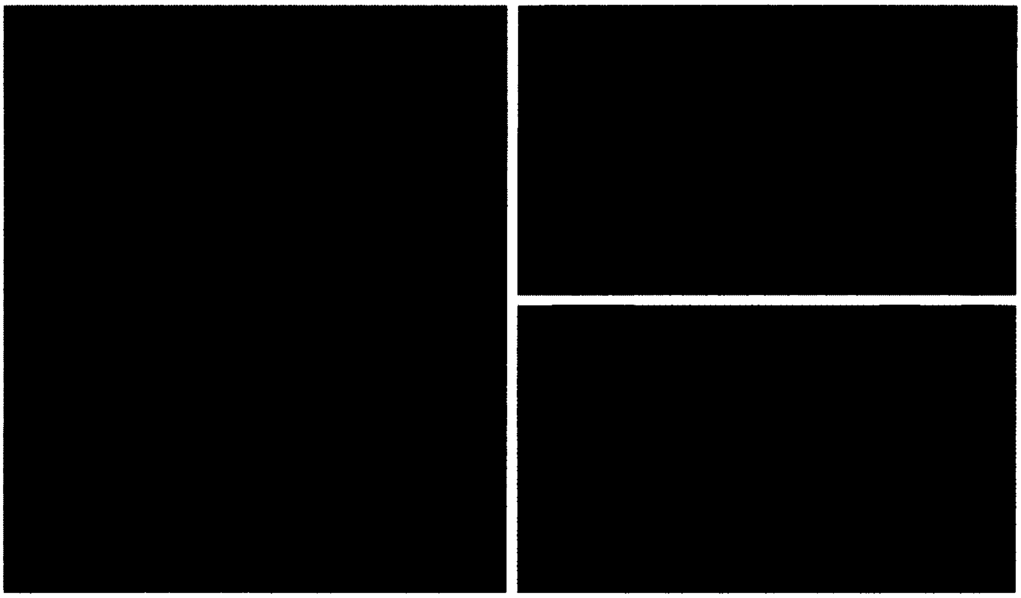


Figure 5.5: From left to right: target structures for the 3DP experiments corresponding to 3DPE1 to 3DPE5 respectively (with perspective, top, and front views).

Therefore, it was appropriate to determine if the information encoded in the components was sufficient to achieve the target structures by self-assembly.

5.3.1 Level One: Rule Set for 3DP Experiments

Similar to the 2DP experiments, programmed component sets followed the same scheme, with the additional difficulty in verifying if the correct rotational information was specified. Random component sets were created by specifying component types using selected information (I to T, and –) assigned to each information location. However in this case, the quantity of each component type in both the programmed and random components sets were multiplied by three to create the maximum number of target structures for each level two and level three 3DP experiment. Component rules and the quantity of each component type are provided in Table 5.4, and the system rules from Table 4.7 (page 101) applied to both groups.

3DPE1	P	9 X (I_1,-,-,-,-,-), 3 X (J_1,-,-,J_1,J_1,-)
	R	9 X (O_3,-,P_1,-,J_1,-), 3 X (S_2,-,-,T_4,-,-)
3DPE2	P	9 X (K_1,-,-,-,-,-), 3 X (M_1,-,-,-,L_1,-), 3 X (N_2,L_1,-,L_1,-,-)
	R	9 X (-,I_1,-,K_1,K_1,-), 3 X (-,-,-,K_1,-,O_3), 3 X (N_2,-,-,T_2,R_4,-)
3DPE3	P	6 X (J_1,-,-,-,-,-), 3 X (P_1,-,-,-,-I_1), 3 X (O_1,-,-,I_1,-,-)
	R	6 X (-,-,P_1,N_1,S_2,-), 3 X (-,I_1,M_2,C_1,-,-), 3 X (O_4,-,S_1,T_3,-,-)
3DPE4	P	6 X (L_1,-,-,-,-,-), 3 X (Q_1,-,-,-,K_1,-), 3 X (R_1,-,-,-,K_1,-)
	R	6 X (J_1,-,-,-,-,Q_4), 3 X (-,-,-,P_1,R_1,M_2), 3 X (O_4,-,-,I_1,-,N_1)
3DPE5	P	6 X (J_1,-,-,-,-,-), 3 X (T_1,-,-,-,-,I_1), 3 X (S_1,-,-,-,I_1,-)
	R	6 X (Q_3,Q_4,-,-,I_1,-), 3 X (-,-,-,K_1,-,-), 3 X (P_4,N_2,-,S_1,-,-)

Table 5.4: Programmed and random (P/R) components sets (represented as '# × (Top, Left, Bottom, Right, Front, Back)', where # represents quantity and the directions refer to component information) for the 3DP experiments (EX)

5.3.2 Level Two: Virtual Execution of Rule Set for 3DP Experiments

The 3DcTAM (*section 4.2: Level Two: Virtual Execution of Rule Set*) was used to evaluate the ability of component rule set (programmed or random) to create its respective target structures.

Level Two: Experimental Setup for 3DP Experiments

The component sets from Table 5.4 were mapped to an abstract representation for the 3DcTAM. Each component's shape was a unit cube. The size of the environment was represented as 4×4×4 units (width×depth×height). These environment dimensions are related to the physical dimensions of the environment as a ratio between the physical components and environment, and rounded down to the nearest unit. Since the 3DcTAM selects tiles/substructures at random to step through the self-assembly process, a different random seed was used to initialize the 3DcTAM for each trial. Ten trials were conducted for each experiment.

Level Two: Experimental Results for 3DP Experiments

Each programmed component set successfully created five of their applicable target structures. These results show that even with a seed component (tile), it is still possible to create multiples of the same target structure, when appropriate component information is used. In contrast, none of the random component sets successfully created at least one target structure, in each 3DP experiment.

The reasons for the unsuccessful random component sets include components having uncomplimentary information within their component sets (3DPE2 and 3DPE5), component sets not being able to consistently create structures due to rotational information (3DPE3 and 3DPE4), and components creating structures consisting of at most two components (3DPE1).

Fisher's Exact Test (one-sided) for analyzing binary data was used to analyze the

3DPE1	Programmed	10	0	0
	Random	0	10	0
3DPE2	Programmed	10	0	0
	Random	0	10	0
3DPE3	Programmed	10	0	0
	Random	0	10	0
3DPE4	Programmed	10	0	0
	Random	0	10	0
3DPE5	Programmed	10	0	0
	Random	0	10	0

Table 5.5: The number of successful and unsuccessful trials for each programmed and random 2DP experiment (2DPE1 - 2DPE3) at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data

results of the level two 3DP experiments (Table 5.5). The results are statistically significant, with a p-value of 0. These successful results provide evidence to support the programming experiments hypothesis at level two in the context of the 3DP experiments.

5.3.3 Level Three: Physical Realization of Rule Set for 3DP Experiments

With the success of each system using a programmed component set at level two, a level three translation was performed to test if the translated component set of each 3DP experiment could self-assemble its respective target structures. A level three translation was not performed on the systems using a random component set, since they were not successful.

Level Three: Experimental Setup for 3DP Experiments

For each experiment trial, components were randomly placed in a jar of mineral oil, and placed on an orbital shaker (their environment). Each system was shaken for 20 minutes, after which the state of each system was recorded, including observations for: the number of target structures created, the number of matching errors, the number of *rotation errors* (between complementary components), and the number of assembly errors. The detailed physical experimental setup for the 3DP experiments is provided in *section C.3: Experimental Procedure for the 3D Systems*

Level Three: Experimental Results for 3DP Experiments

Each experiment was successful in creating at least one target structure (Figure 5.6). Figure 5.7 (page 120) shows an example of a successful trial from each 3DP experiment. In 3DPEl and 3DPE2, there were no matching and rotation errors (as this was not possible due to the 5-magnetic-bit patterns present), and no assembly errors. Figure 5.8 (page 121) shows the rotation errors for 3DPE3, 3DPE4, and 3DPE5. There were no matching and assembly errors in 3DPE3, 3DPE4, and 3DPE5. As with the 2DP experiments, it was challenge to create a physical environment temperature of one.

Fisher's Exact Test (one-sided) for analyzing binary data was used to determine the statistical significance of creating target structures in each 3DP experiment (Table 5.6). All five programming experiments are statistically significant at the 0.01 level (i.e there is a 99% percent certainty the results are not due to chance). The successful 3DP experiments demonstrate that physical information can be used to enable components to self-assemble into closed 3D target structures with symmetric/asymmetric features, and provide evidence to support the hypothesis for the programming experiments.

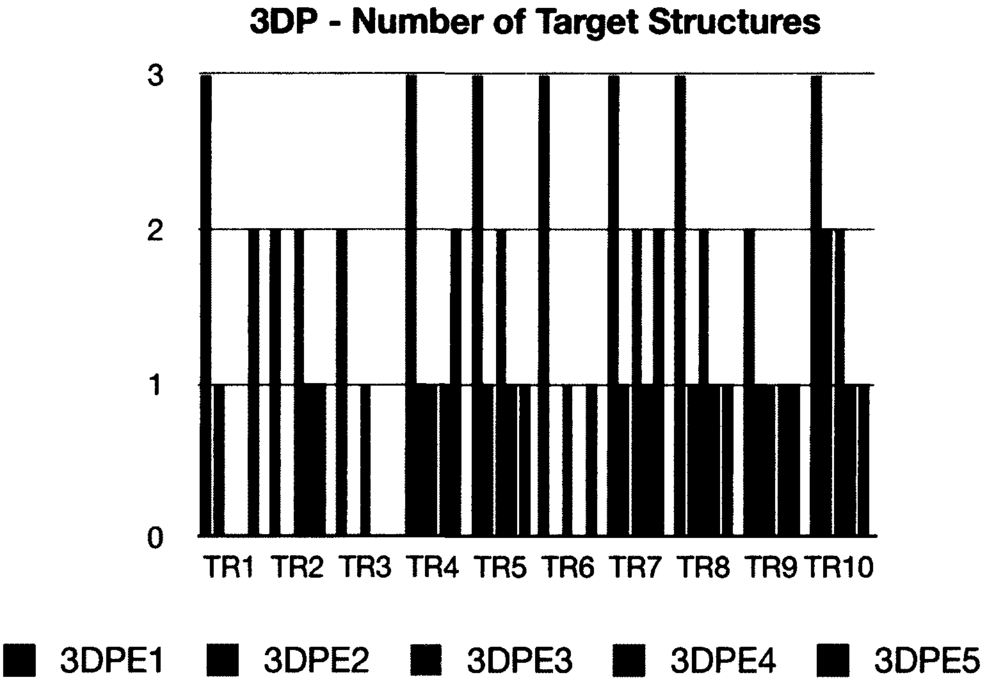
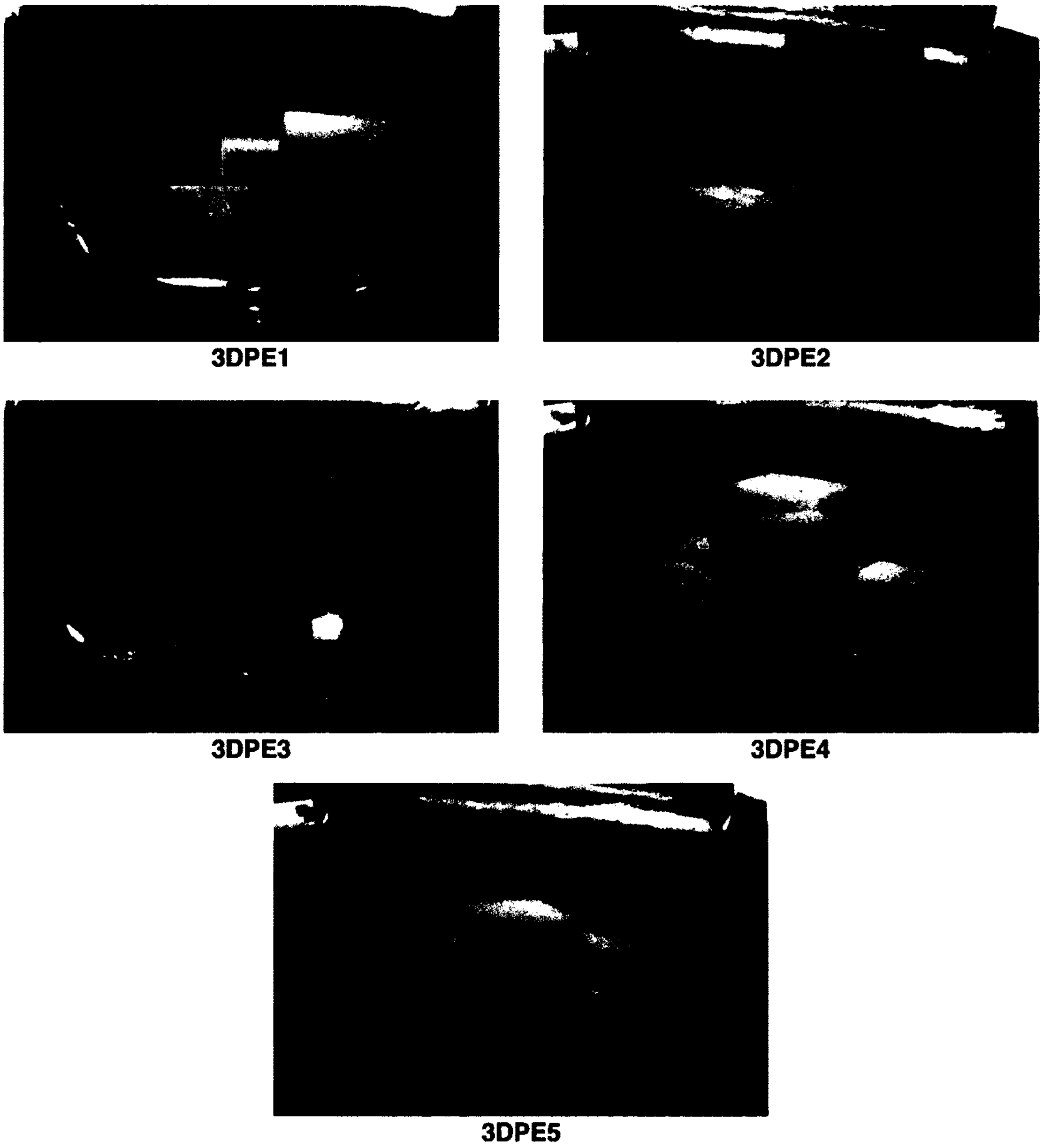


Figure 5.6: Number of target structures created in each of the ten trials (TR1 - TR10), for each of the five programming experiments (3DPE1 - 3DPE5) at level three.

3DPE1	Programmed	10	0	0
3DPE2	Programmed	7	3	0.002
3DPE3	Programmed	9	1	0
3DPE4	Programmed	7	3	0.002
3DPE5	Programmed	9	1	0

Table 5.6: Number of successful and unsuccessful trials for the programming experiments (PEX1 - PEX5) at level three, with corresponding p-values (calculated using Fisher's Exact Test (one-sided) for analyzing binary data, with respect to 0 successful and 10 unsuccessful trials for the random sets for each programming experiment from their level two results)



3DPE1

3DPE2

3DPE3

3DPE4

3DPE5

Figure 5.7: Photographs of successful level three programming experiment trials (PEX1 - PEX5).

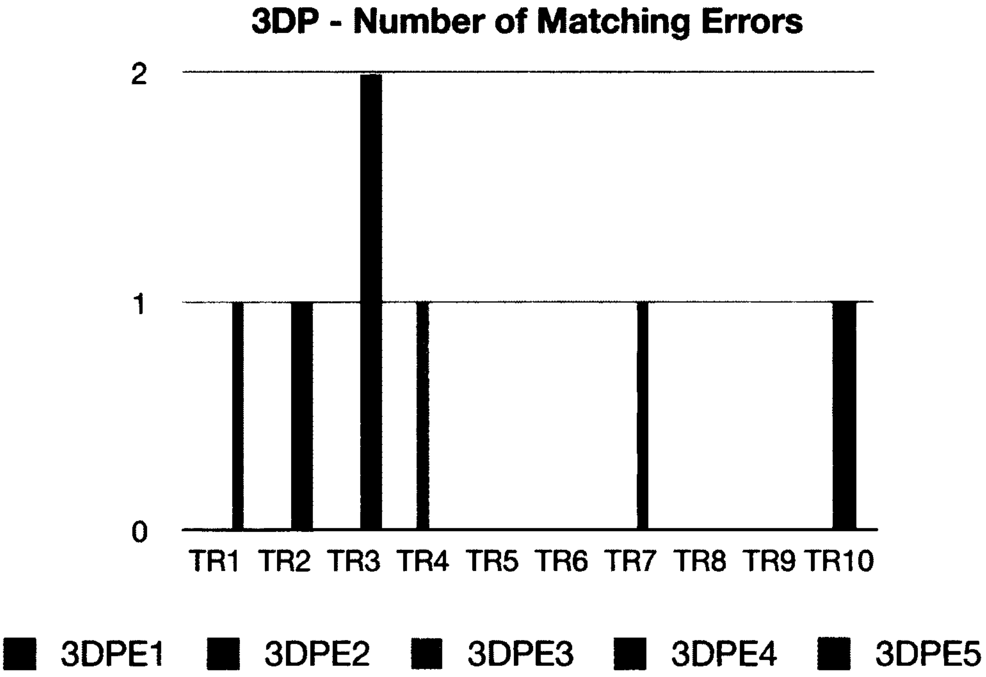


Figure 5.8: Number of rotation errors in each of the ten trials (TR1 - TR10), for each of the five programming experiments (PEX1 - PEX5) at level three.

5.4 Summary

The first set of experiments using the three-level approach were presented in this chapter. Here the three-level approach was used directly in the context of a programming paradigm, where the output of a *self-assembly program* is a target structure. Three 2DP and five 3DP experiments were conducted, each with a specific target structure. These experiments were used to test the programming experiments hypothesis of wether physical information can be used to enable components to self-assemble into closed target structures with symmetric/asymmetric features. The independent variable in these experiments was the set of components (programmed or random), and the dependent variable was the resulting set of self-assembled structures. All the types of physically encoded information, as part of the second 2D and 3D physical information encoding schemes, Table 4.5 (page 97) and Table 4.7 (page 101) respectively, were used across the 2DP and 3DP experiments. All the programming experiments are statistically significant with a p-value of 0 or 0.002. The successful results of the 2DP and 3DP experiments provide evidence to support the hypothesis for the programming experiments.

Furthermore, the successful programming results provide credibility to the three-level approach, including the set of rules used at level one, and the 2DcTAM and the 3DcTAM used at level two (concurrent self-assembly, where no seed tiles are required and component rotations are permitted, at temperature one). With the success of the second 2D and 3D physical information encoding schemes, further experiments extending the three-level approach are presented in *Chapter 6: Evolving Self-Assembly Rule Sets* and *Chapter 7: Staging the Self-Assembly Process.*

Chapter 6

Evolving Self-Assembly Rule Sets

With the success of the programming experiments (presented in *Chapter 5: Programming Self-Assembling Systems*), the next step was to demonstrate how to automatically generate sets of rules using computer software to design physical self-assembling systems via physically encoded information. As the sophistication of self-assembling systems continues to increase, it will be more challenging to use top-down design due to self-assembly being an algorithmically NP-complete problem (*section 2.2.5: The Complexity of Self-Assembly*). Evolutionary computing is well-suited for addressing such problems (Mitchell 1996). By incorporating evolutionary computing into the three-level approach (Figure 6.1), it couples bottom-up construction (self-assembly) with bottom-up design (evolution). As a corollary, an evolutionary methodology is beneficial in that no knowledge of a target structure's morphology is required, only it's functionality.

Two 2D evolutionary (2DE) experiments and three 3D evolutionary (3DE) experiments were conducted. The evolutionary algorithm (including genotype and phenotype representations, multi-objective fitness function, and selection, crossover, and genetic operators) is detailed, followed by the 2DE and 3DE experiments. This research and the 2DE and 3DE experiments have been published in Bhalla et al. (2010) and Bhalla et al. (2011a) respectively.

6.1 Evolutionary Algorithm

The objective of the evolutionary algorithm is to search for a *good enough solution*, i.e. a component set (type and concentration) able to self-assemble into a single target

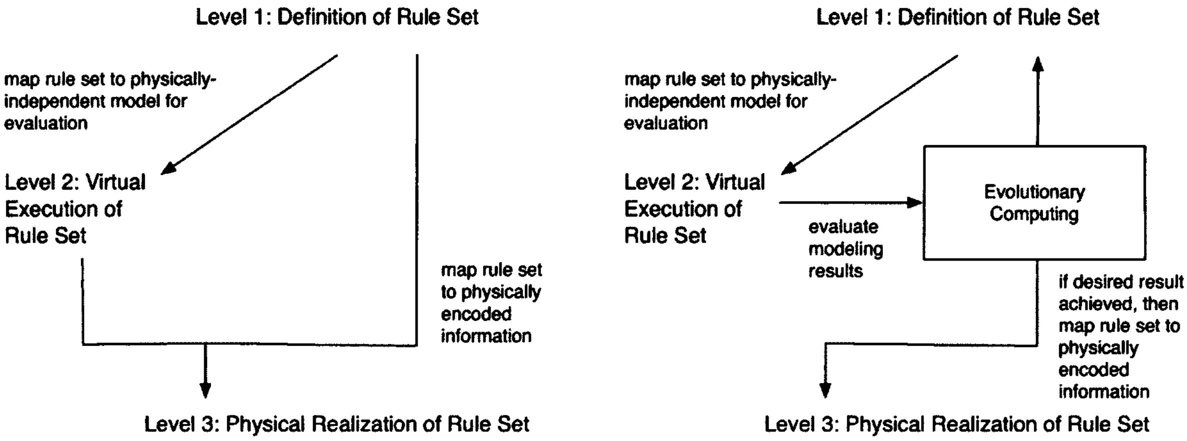

Level 1: Definition of Rule Set
map rule set to physically-independent model for evaluation
Level 2: Virtual Execution of Rule Set
map rule set to physically encoded information
Level 3: Physical Realization of Rule Set
Level 1: Definition of Rule Set
map rule set to physically-independent model for evaluation
Level 2: Virtual Execution of Rule Set
evaluate modeling results
Evolutionary Computing
if desired result achieved, then map rule set to physically encoded information
Level 3: Physical Realization of Rule Set

Figure 6.1: Three-level approach (left) incorporating evolutionary computing (right).

structure. Environment and system (fits and breaks) rules are fixed. A generational evolutionary algorithm (Mitchell 1996) is used. The evolutionary unit, *gene*, is a single component (2D or 3D). A collection of gene sequences, *databank*, is used to identify and compare genes. A linear sequence constitutes a single gene (Top, Left, Bottom, Right) representing a 2D component (using A to H to indicate information orientation on an edge, and the – symbol to indicate a neutral site), and (Top, Left, Bottom, Right, Front, Right) representing a 3D component (using I to T with subscripts 1 to 4 to indicate information orientation on a face, and the – symbol). There are 6,561 total and 1,665 unique 2D genes (when considering 2D shape and rotation), and there are 1,291,467,969 total and 53,977,737 unique genes (when considering three-dimensional shape and rotations). By varying the set of genes (representing a set of components in a system) and the information (capital letter) associated with each gene, both are the evolvable elements, different phenotypes can be created (resulting set of structures in a system at the end of the self-assembly process).

Elitism was used, where a portion of the individuals with highest fitness were copied to the next generation. The parameter settings for the number of generations, population

[illegible]	[illegible]	[illegible]
Number of Generations	5,000	10,000
Population Size	50	100
Elitism	10%	10%

Table 6.1: Evolutionary algorithm parameter settings for 2D and 3D systems.

size, and elitism for the 2D and 3D evolutionary experiments are provided in Table 6.1. The following is an overview of the genotype and phenotype representations, fitness function, and selection, crossover, and genetic operators used.

6.1.1 Genotype and Phenotype Representations

Genotype and phenotype representations are the same for both the 2D and 3D case. An individual's genotype representation is a variable length list of genes (Figure 6.2). At least two genes define a genotype (since this is the minimum for self-assembly to occur). An individual's phenotype representation is the resulting set of self-assembled structures. A single genotype representation may have more than one phenotype representation, depending on the set of components and assembly steps. As a worst-case example, a genotype that consists of n components (where $n \in \mathbb{N}$) with information I on all faces and n components with information J on all faces would result in at most $2n!$ phenotypes. Consequently, it is not practical to test all the resulting phenotypes corresponding to a large genotype. Therefore, each individual (genotype) is evaluated three times, at each generation, to help determine the fitness of an individual.

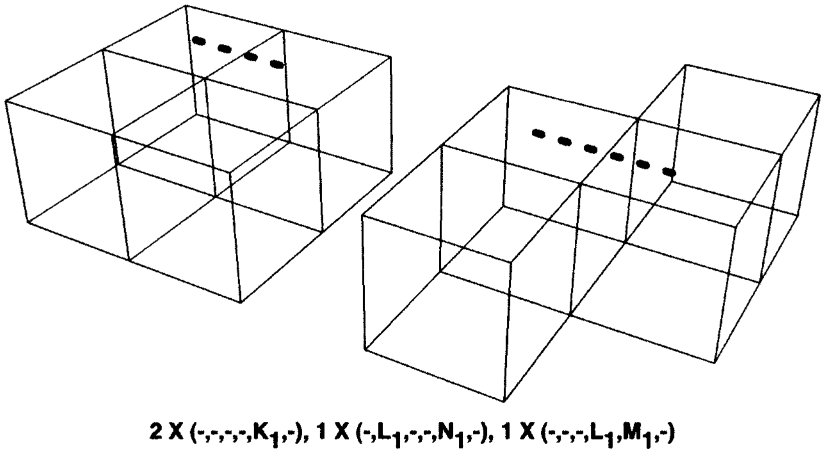

2 X (-,-,-,-,K_1,-), 1 X (-,L_1,-,-,N_1,-), 1 X (-,-,-,L_1,M_1,-)

Figure 6.2: Example of a 3D genotype that has more than one phenotype, where the 180° bond between information M and N can assemble in two different configurations (shown with the dashed lines).

6.1.2 Multi-Objective Fitness Function

Two multi-objective fitness functions are used to evaluate each 2D and 3D individual. Both multi-objective fitness functions consist of a *general solution* and a *refined solution.* The general solution describes the morphology of a target structure. A refined solution is one that minimizes the number of remaining open assembly locations (preference for closed structures) and potential for assembly errors (due to magnetic interactions). Next, the objectives and functions comprising the 2D and 3D multi-objective fitness functions are provided.

2D Multi-Objective Fitness Function

Seven objectives can be categorized into evaluating a *general* and a *refined* solution (Figure 6.3, page 128). The general solution has five objectives: (1) area (A), (2) perimeter (P), (3) Euler (E), (4) *z-axis*, and (5) *matches.* Each of these objectives is used to achieve the *shape* of the target structure. The area, perimeter, and Euler (*connectivity* of a shape) are calculated using 2D Morphological Image Analysis (Soille 2003), and are

0			
5	2		
5	4	2	
4	3	3	4

Table 6.2: A sliding window technique is used (matrix) as a sum of magnetic errors (odd number of magnets must match at each position along the sliding window) and is applied to all potential two-component key-to-key interactions in a system

provided in Equations 6.1 - 6.3 (where n_s is the number of squares (components), n_e is the number of edges, and n_v is the number of edges). The second-moment of inertia in the z-axis (Beer et al. 2009) is calculated to identify similar, but rotated structures. To distinguish between reflected structures (which are not permitted), the number of matching components between a self-assembled structure and the target structure is calculated. A refined solution is accounted for by using two objectives: (6) *locations* and (7) *error*. We consider a refined solution as one that minimizes the number of remaining open assembly locations and potential assembly errors (due to magnet interactions). The combination of these two objectives also reduces the number of unique components required.

$$A = n_s \tag{6.1}$$

$$P = -4 + 2n_e \tag{6.2}$$

$$E = n_s - n_e + n_v \tag{6.3}$$

Each objective is normalized, using the highest and lowest values from a generation (Bentley & Wakefield 1997). For objectives one to five (i), the average normalized objective (ANO_i) over three 2DcTAM evaluations is calculated and compared to the target

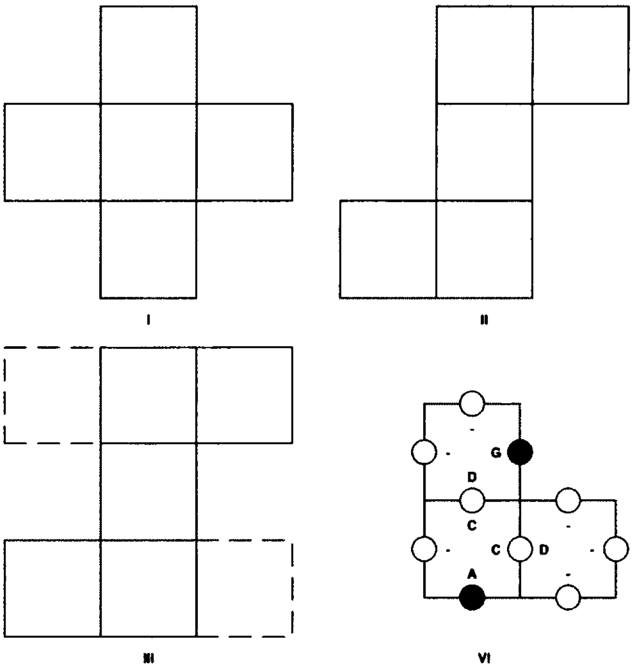


Figure 6.3: Fitness objective examples: structure I ($A = 5$, $P = 12$, and $E = 1$); structure II has the same second moment of inertia for its reflected equivalent; number of matches between reflected structure II is 3 (III); number of open locations is 2 (black circles, IV); an error of 2 in IV (Table 6.2).

objective (TO_i) value. For objective six, the normalized average over the three 2DcTAM evaluations (ANO_6) is calculated. For objective seven, the normalized objective (NO_7) is calculated with respect to a genotype. The objectives are then weighted to give the final fitness score F^{2D} (Equation 6.4). The weights were selected by conducting preliminary experiments.

$$F^{2D} = (0.18\sum_{i=1}^{5}|TO_i - ANO_i|) + 0.05(ANO_6 + NO_7) \tag{6.4}$$

3D Multi-Objective Fitness Function

Eight objectives can be categorized into evaluating a general and a refined solution (Figure 6.4). The general solution has six objectives: (1) volume (V), (2) surface area (S), (3) mean breadth (B), (4) Euler (E), (5) *z-axis*, and (6) *matches*. The volume, surface area, mean breadth, and Euler are calculated using Equations 6.5 to 6.8 (where n_c is the number of cubes (components), n_f is the number of faces, n_e is the number of edges, and n_v is the number of vertices). Together, these six general objectives are sufficient to describe the 3D shape of the target structures considered in the experiments. The volume, surface area, integral mean curvature, and Euler (connectivity of a shape) are calculated using 3D Morphological Image Analysis (Blasquez & Poiraudeau 2003; Michielsen & de Raedt 2000; Soille 2003). The second moment of inertia in the z-axis (Beer et al. 2009) is calculated to identify either identical structures or different structures which have similar reflected features. To distinguish between reflected structures, the number of matching components between a self-assembled structure and the target structure is calculated. A refined solution is accounted for by using two objectives: (7) *locations*, and (8) *error*. A refined solution is considered as one that minimizes the number of remaining open assembly locations (for the creation of closed target structures), and potential assembly errors (due to magnetic interactions). Errors in magnetic interactions is applied to all

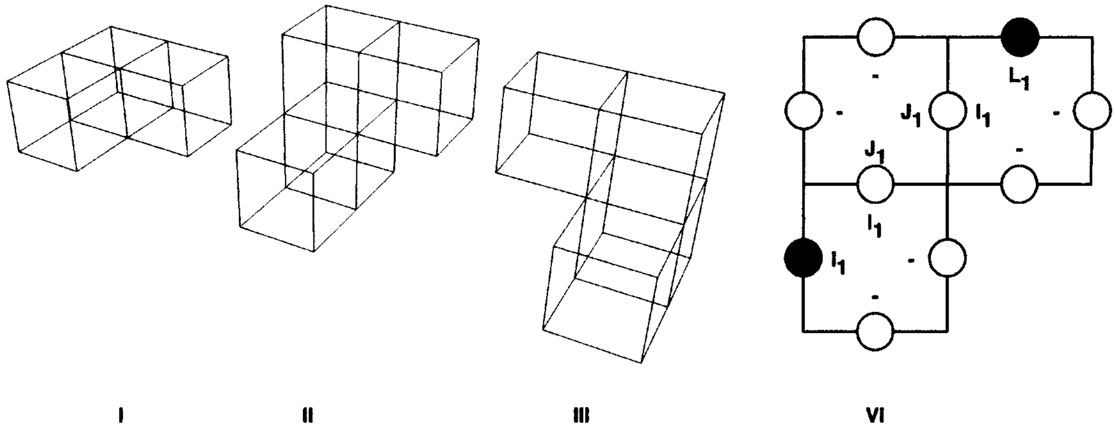


Figure 6.4: Fitness objective examples: structure I (V = 3, S = 14, B = 2.5, and E = 1), structures II and III have the same moment of inertia, the number of matches between structures II and III which have similar reflected features is 3, the number of open locations is 2 (indicated using black circles in IV; 2D top view), and the potential error score in IV is 8 (Table 6.3).

potential two-component key-to-lock and key-to-key (all five magnets positions are considered in key-to-key errors, and partial interactions are not accounted for) in a system. The potential magnet error is calculated as the sum of the scores in Table 6.3, where each cell in the matrix is the sum of errors occurring between two pieces of information in all four orientations. The combination of these two objectives also reduces the number of unique components required, as well as favouring 5-magnetic-bit patterns with higher rotational freedom.

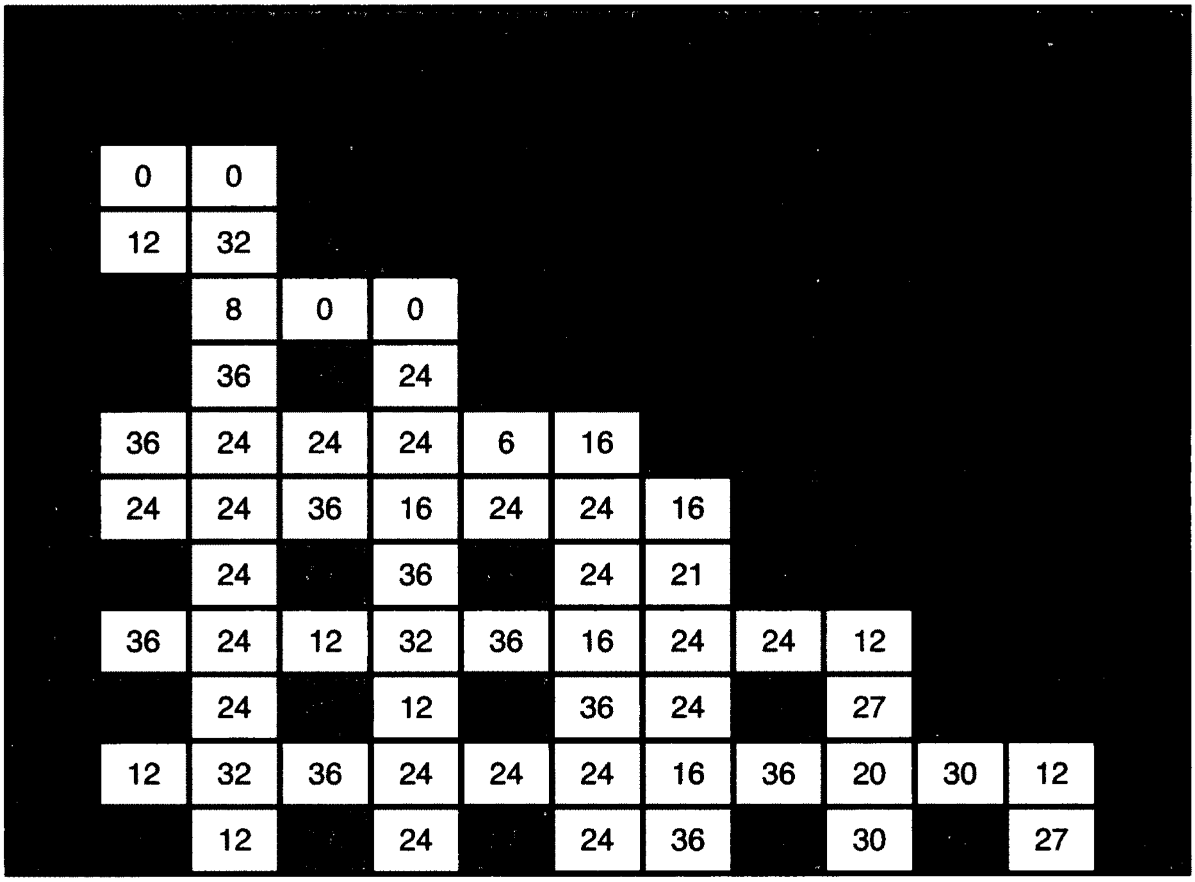
$$V = n_c \tag{6.5}$$

$$S = -6n_c + 2n_f \tag{6.6}$$

$$B = (3n_c - 2n_f + n_e)/2 \tag{6.7}$$

$$E = -n_c + n_f - n_e + n_v \tag{6.8}$$

Each objective is normalized, using the highest and lowest fitness scores from the



0	0									
12	32									
	8	0	0							
	36		24							
36	24	24	24	6	16					
24	24	36	16	24	24	16				
	24		36		24	21				
36	24	12	32	36	16	24	24	12		
	24		12		36	24		27		
12	32	36	24	24	24	16	36	20	30	12
	12		24		24	36		30		27

Table 6.3: Magnetic error interactions matrix (in reference to Table 4.7, page 101), where the numbers are the sum of all errors between information in the four orientations (using the number of magnets, i.e. two or one, in each mismatched location), and where N/A is in reference to lock-to-lock interactions not being possible

current generation (Bentley & Wakefield 1997). This method has been shown to be an effective way to calculate multi-objective fitness (Corne & Knowles 2007). For objective i (where i varies from 1 to 6) the average normalized objective (ANO_i) over three 3DcTAM evaluations is calculated and compared to the target objective (TO_i) value. For objective seven, the normalized average over the three 3DcTAM evaluations (ANO_7) is calculated. For objective eight, the normalized objective (NO_8) is calculated with respect to a genotype. The average is not used with objective 8 as the error calculated is based exclusively on a genotype, and does not vary over the three 3DcTAM evaluations. The objectives are then weighted and summed to give a final fitness score F^{3D} (Equation 6.9). The weights were selected from preliminary experiments, and based on the 2D evolutionary experiments conducted. It was found that a weighting where the refined objectives accounted for more than 10% of the overall fitness function did not lead to good solutions.

$$F^{3D} = (0.15\sum_{i=1}^{6}|TO_i - ANO_i|) + 0.05(ANO_7 + NO_8) \tag{6.9}$$

6.1.3 Selection, Crossover, and Genetic Operators

The same selection, crossover, and genetic operators are used in both the 2D and 3D case. The fitness scores for each individual are used during selection. Roulette-wheel selection is used to select two parents (favouring lowest fitness scores). The two parents, using a variable-length crossover operator, are used to create two children. Each common gene (determined by the gene databank) between the two parents is copied to each child. Each uncommon gene, for example the gene from parent one, has a 90% probability of being copied to child one (likewise for parent two and child two). After crossover is performed to create two children, the genetic operators duplication, deletion, and mutation are applied to each child. There is a 10% probability of a single gene, chosen at random, of being duplicated, and likewise being deleted. For each information location in a gene, there is a 10% probability of being mutated (equal probability A to H, and – in the 2D case, and equal probability I to T in all four orientations, and – in the 3D case).

6.2 Hypothesis Statement

The hypothesis for the evolutionary experiments was,

Hypothesis: given a closed target structure with symmetric/asymmetric features, an evolutionary algorithm can be used to evolve a set of components that are able to self-assemble into the desired target structure.

The three-level approach with the addition of evolutionary computing (evolved rule sets at level one) was used to test this hypothesis, virtually (level two) and physically (level three). Two 2DE experiments and three 3DE experiments (referred to as 2DEE1, 2DEE2, and 3DEE1 to 3DEE3 respectively) were conducted to test this hypothesis.

A target structure was assigned to each experiment (provided in *section 6.3: 2D Evolutionary Experiments and Results* and *section 6.4: 3D Evolutionary Experiments and Results*). As with the programming experiments, enough components were supplied to create up to three 2D and 3D target structures. Again, the number of components corresponds to the capacity of the corresponding environments. Ten trials were conducted for each experiment[1]. Like the programming experiments, a trial was evaluated by be successful if all three 2D and 3D target structures were created at level two, and if at least one 2D and 3D target structures were created at level three. The independent variable in these experiments was the set of components, defined by their type and frequency.

[1] An additional five trials for each 2D experiment, to the original five trials presented in (Bhalla et al. 2010), were conducted to collect further data.

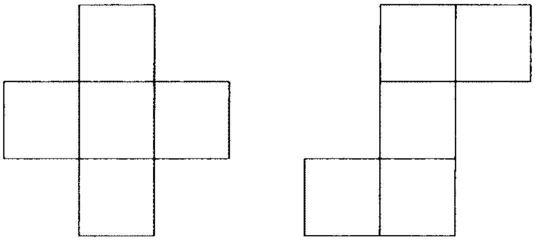


Figure 6.5: Target structures for 2DEE1 (left) and 2DEE2 (right).

The dependent variable is the resulting self-assembled structures. For each experiment, an *evolved* component set was specified along with a *random* component set to test the independent variable.

6.3 2D Evolutionary Experiments and Results

Two 2DE experiments were conducted, where a target structure was assigned to each experiment (Figure 6.5). The experimental procedures and results are described according to the three-level approach: evolving rule sets (level one), modelling using 2DcTAM (level two), and physically testing systems using the second 2D physical information encoding scheme (level three).

6.3.1 Level One: Rule Set for 2DE Experiments

The evolutionary algorithm used 5,000 generations, with a population size of 50 individuals, for each run. The initial individual (genotype) length was set to the required number of components to create one target structure. Figure 6.6 shows the evolutionary algorithm results. Five runs were conducted for each experiment. For 2DEE1, the two optimal solutions were achieved. The second solution was chosen for these experiments, as components from previous 2DPE1 could be reused. For 2DEE2, the single optimal solution was achieved. For the random component sets, components were created by se-

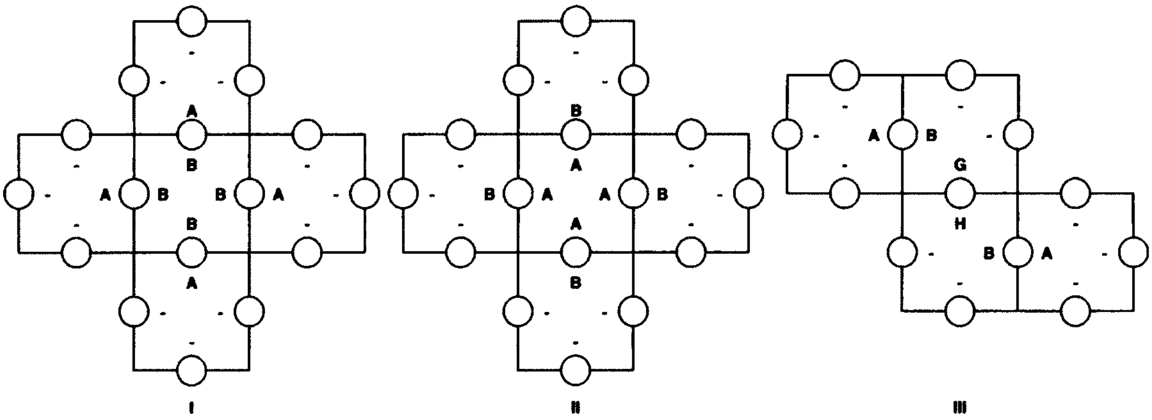


Figure 6.6: The two optimal solutions for 2DEE1 (I and II) and the single optimal solution for 2DEE2 (III).

lecting, with uniform probability, the information assigned to each location. The number of components randomly generated were equal to the required number of components to create one target structure. A summary of the component rules, for each experiment, is provided in Table 6.4. The number of components specified (evolved and random) were multiplied by three in order to create the maximum number of target structures for the 2DE experiments. The system rules from Table 4.5 (page 97) were applicable to both groups.

6.3.2 Level Two: Virtual Execution of Rule Set for 2DE Experiments

The 2DcTAM was used to virtually evaluate the ability of each rule set (evolved or random) to create its respective target structure. Although the 2DcTAM is used by the evolutionary algorithm, it is used again to verify the creation of multiple target structures concurrently for each rule set (evolved and random).

Level Two: Experimental Setup for 2DE Experiments

The component rules from Table 6.4 were mapped to an abstract representation for the 2DcTAM. The same parameter settings for the 2DcTAM from *section 5.2.2: Level*

[redacted]	[redacted]	[redacted]
2DEE1	E	3 X (A,A,A,A), 12 X (-,B,-,-)
	R	3 X (-,-,B,G), 3 X (-,D,E,E), 3 X (C,-,-,C), 3 X (C,E,-,-), 3 X (-,F,B,H)
2DEE2	E	3 X (-,B,G,-), 6 X (-,-,-,A), 3 X (H,-,-,B)
	R	3 X (G,H,H,-), 3 X (-,A,-,-), 3 X (-,H,-,-), 3 X (-,-,E,A)

Table 6.4: Evolved and random (E/R) component sets (represented as '# × (Top, Left, Bottom, Right)', where # represents quantity and the directions refer to component information) for the 2DE experiments (EX)

Two: Experimental Setup for 2DP Experiments was used: each component's shape was a unit square, the environment boundary size was 10×10 units, and the environment temperature was set to one. Again, a different random seed was used to initialize the 2DcTAM for each of the ten trials.

Level Two: Experimental Results for 2DE Experiments

Each evolved component set successfully created three of their applicable target structures, at level two. These results show that even without a component acting as a seed, it is still possible to successfully create target structures. Furthermore, these results show that it is possible to create multiples of the same target structure, when appropriate component information is used. In contrast, none of the random component sets successfully created at least one target structure, in each trial. In this case, the same reason for the unsuccessful results applies to both random sets. For 2DEE1, the first and last component types will form substructures that are independent from substructures formed by the second, third, and fourth component types. Likewise for 2DEE2, the first and third component types will form substructures that are independent from substructures formed by the second and fourth component types.

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2DEE1	Evolved	10	0	0
	Random	0	10	
2DEE2	Evolved	10	0	0
	Random	0	10	

Table 6.5: The number of successful and unsuccessful trials for each evolved and random evolutionary experiment (2DEE1 and 2DE2) at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data

To evaluate the level two results, Fisher's Exact Test (one-sided) for analyzing binary data was used. Each experiment is statistically significant with a p-value of 0 (Table 6.5). Therefore, the hypothesis for the evolutionary experiments is supported at level two by the evidence from these successful 2DE experiments.

6.3.3 Level Three: Physical Realization of Rule Set for 2DE Experiments

With the success of each system using an evolved component set at level two, a level three translation was performed to test if the translated component set of each system could self-assemble into its respective target structure. A level three translation was not performed on the systems using a randomly generated component set, since they were not successful.

Level Three: Experimental Setup for 2DE Experiments

Component mapping followed Table 4.5 (page 97). The physical experimental procedure followed *section 5.2.3: Level Three: Experimental Setup for 2DP Experiments.* Randomly placed components on the surface of a tray were shaken for 20 minutes, after which the state of the system was recorded: number of target structures, number of matching errors,

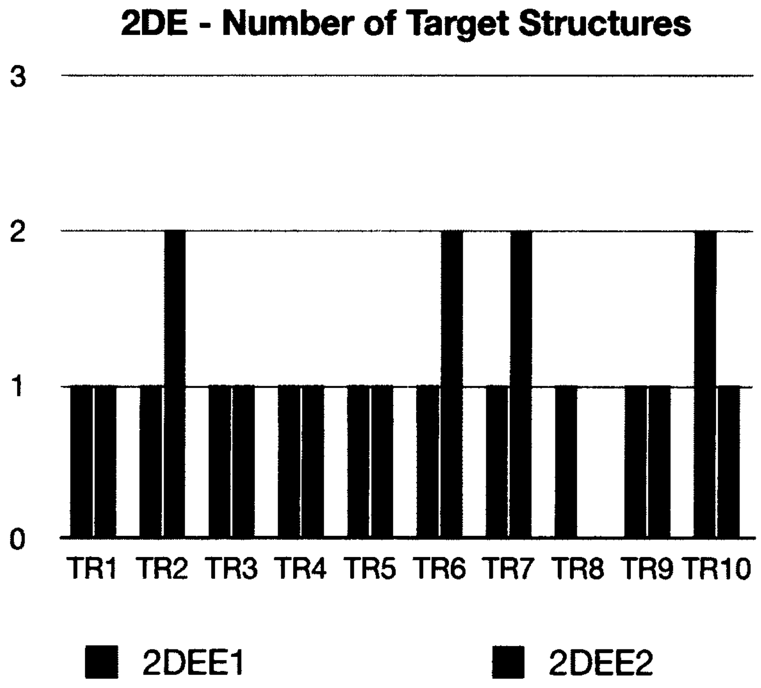
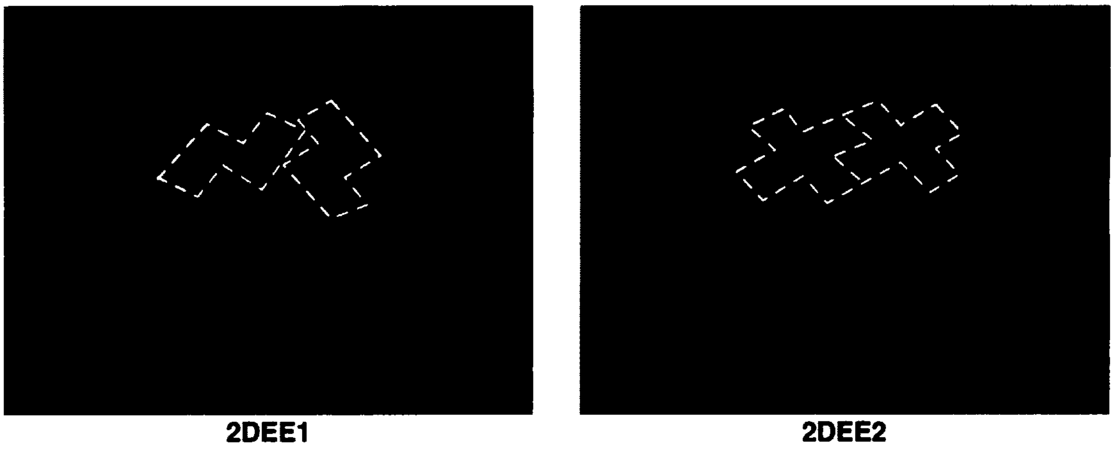


Figure 6.7: Number of target structures created in each of the ten trials (TR1 - TR10), for each of the two evolutionary experiments (2DEE1 and 2DEE2) at level three.

and number of assembly errors. Details regarding the physical experiment setup for the 2DE experiments is provided in *section C.2: Experimental Procedure for the 2D Systems.*

Level Three: Experimental Results for 2DE Experiments

Each trial, for each experiment, was successful in creating at least one target structure. Figure 6.7 shows the number of target structures achieved in each trial. Figure 6.8 shows the final state for the best trial for each experiment. In 2DEE1, there were no matching errors (as this was not possible due to the 3-magnetic-bit codes present) and no assembly errors. In 2DEE2, there was only one matching error (trial five) and no assembly errors. As structures self-assembled, the environmental free space was reduced, constraining the rotation of substructures and sometimes constraining single components from reaching assembly locations.



2DEE1

2DEE2

Figure 6.8: Photographs of successful 2DE experiments (2DEE1 and 2DEE2), where adjacent components to the outlined shapes are not attached to the self-assembled structures (resting positions when energy was removed from the environment).

Fisher's Exact Test (one-sided) for analyzing binary data was used to determine the statistical significance of the results from the 2DE experiments. For both 2DE experiments, the p-value is 0, which is considered to be statistically relevant (Table 6.6). As a result, these successful results provide evidence to support the hypothesis that given a closed 2D structure with symmetric/asymmetric feature, an evolutionary algorithm can be used to evolve a set of components that are able to self-assemble into the desired target target structure.

6.4 3D Evolutionary Experiments and Results

Three 3DE experiments were conducted (Figure 6.9, page 141). As with the 3DP experiments, these target structures vary in the number of components and their symmetric/asymmetric features, and cannot be created using the underlying cube lattice pattern. As a result, these target structures are appropriate to test if the information encoded in the component sets is sufficient to achieve the target structure using self-assembly. The experimental procedures and results are described according to the three-level approach:

2DPE1	Programmed	10	0	0
2DPE2	Programmed	9	1	0

Table 6.6: Number of successful and unsuccessful trials for the evolutionary experiments (2DEE1 and 2DEE2) at level three, with corresponding p-values (calculated using Fisher's Exact Test (one-sided) for analyzing binary data, with respect to 0 successful and 10 unsuccessful trials for the random sets for each evolutionary experiment from there level two results)

evolving a component rule set (level one), modelling using the 3DcTAM, and physically testing systems using the second 3D physical information encoding scheme (level three).

6.4.1 Level One: Rule Set for 3DE Experiments

Although multiple solutions were evolved for each 3DP experiment, the selected solutions were based on those which used component designs previously used in the 3DP experiments (to reduce financial costs of fabricating components). The number of components specified (evolved and random) were multiplied by three in order to increase the chances of being able to create up to three examples of the target structures. This method for creating the evolved and random component sets was chosen as part of the experimental setup with the aim of providing proof-of-concept evidence for the evolutionary paradigm, by conducting a statistical significance tests. The component sets used in the evolutionary experiments is provided in Table 6.7. System rules from Table 4.7 (page 101) applied to both evolved and random component sets.

6.4.2 Level Two: Virtual Execution of Rule Set for 3DE Experiments

The 3DcTAM was used to evaluate the ability of each rule set (evolved or random) to create its respective target structure. Although the 3DcTAM is used by the evolutionary

Figure 6.9: From left to right: target structures for the 3DE experiments corresponding to 3DEE1 to 3DEE3 respectively (with perspective, top, and front views).

3DEE1	E	3 X (-,I_1,-,I_1,-,-), 6 X (-,-,J_1,-,-,-)
	R	3 X (-,-,M_2,-,I_1,I_1), 6 X (L_1,P_1,-,-,-,-)
3DEE2	E	12 X (-,-,L_1,-,-,-), 3 X (-,-,-,M_2,K_1,K_1), 3 X (K_1,-,K_1,-,-,N_1)
	R	12 X (I_1,-,I_1,-,S_2,R_3), 3 X (-,S_4,L_1,-,-,-), 3 X(-,-,I_1,L_1,-,-)
3DEE3	E	9 X (-,-,-,-,-,J_1), 3 X (-,S_1,I_1,-,I_1,-), 3 X (I_1,-,-,-,T_4,-)
	R	9 X (-,O_3,N_1,-,N_1,-,), 3 X (-,M_1,J_1,-,-,-), 3 X (-,L_1,-,-,-,M_2)

Table 6.7: Evolved and random (E/R) components sets (represented as '# × (Top, Left, Bottom, Right, Front, Back)', where # represents quantity and the directions refer to component information) for the 3DE experiments (EX)

algorithm, it also used to verify the creation of the three 3D target structures.

Level Two: Experimental Setup for 3DE Experiments

With the exception of component rules following Table 6.7, the same level two experimental setup from the 3DP experiments was used for the level two 3DE experiments (*section 5.3.3: Level Two Experimental Setup for the 3DP Experiments*).

Level Two: Experimental Results for 3DE Experiments

Table 6.8 provides the results of the 3DP experiments at level two. Each evolved component set successfully created all three target structures, whereas the random component sets were unsuccessful in generating any target structures. Along with the level two 3DP experiments, these evolved results further support that seed tiles (components) are not required to create target structures. The causes for the unsuccessful randomly generated results follow the same reason as the level two 3DP experiments of random component sets having uncomplimentary information (3DPEE1 and 3DEE2), and random component sets not being able to consistently create structures due to rotational information (3DEE3). To analyze the evolutionary results, Fisher's Exact Test (one sided) for analyzing binary data was used. The p-value for each 3DP experiment is 0. These successful results provide evidence to support the hypothesis for the evolutionary experiments at level two in the context of the 3DE experiments.

6.4.3 Level Three: Physical Realization of Rule Set for 3DE Experiments

Each system using an evolved component set was successful at level two. As a result, a level three translation was performed to test if the translated component set of each evolved system could self-assemble into its respective target structure. Since the results were unsuccessful, a level three translation was not performed for the random component sets.

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3DEE1	Evolved	10	0	0
	Random	0	10	
3DEE2	Evolved	10	0	0
	Random	0	10	
3DEE3	Evolved	10	0	0
	Random	0	10	

Table 6.8: The number of successful and unsuccessful trials for each evolved and random evolutionary experiment (3DEE1 - 3DEE3) at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data

Level Three: Experimental Setup for 3DE Experiments

The same level three physical experimental setup used in the 3DP experiments (*section 5.3.3: Level Three: Experimental Setup for 3D Experiments*) was used for the level three 3DE experiments, with the exception of jar placements on the orbital shaker (*sectionC.3: Experimental Procedure for the 3D Systems*). At the conclusion of each physical 3DE trial, the state of system was recorded (number of target structures created, matching errors, rotation errors, and assembly errors).

Level Three: Experimental Results for 3DE Experiments

Figure 6.10 shows the results for each 3DP experiment at level three. Figure 6.11 (page 145) shows an example of the final state of a successful trial, for each experiment. Rotation errors did not occur in 3DEE1 and 3DEE2, but did in 3DEE3 (one rotation error in trial one). Matching and assembly errors did not occur in all three 3DE experiments. Fisher's Exact Test (one-sided) for analyzing binary data was used to determine the statistical significance of creating target structures in each experiment (Table 6.9, page

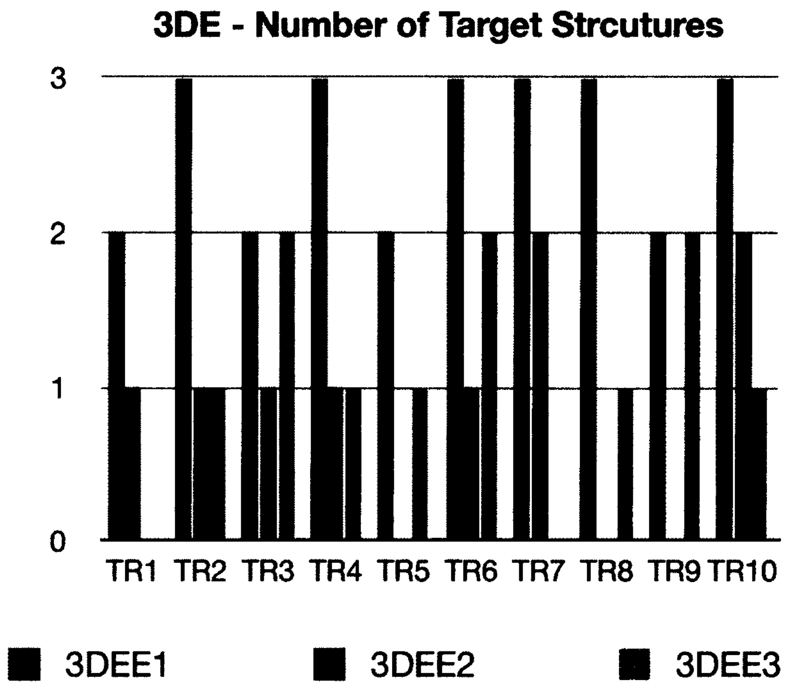
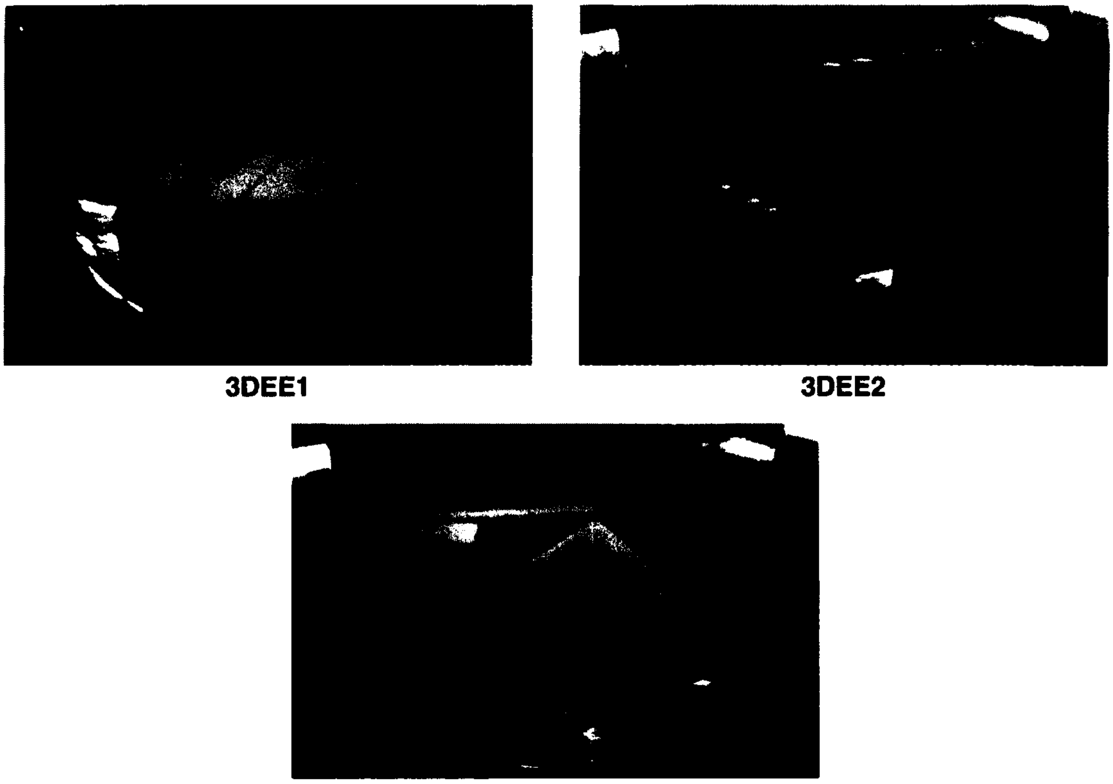


Figure 6.10: Number of target structures created in each of the ten trials (TR1 - TR10), for each of the three evolutionary experiments (3DEE1 - 3DEE3) at level three.

145). All three 3DE experiments are statistically significant at the 0.01 level. These successful 3DP experiments provide evidence to support the evolutionary hypothesis that given the attributes of a closed 3D target structure with symmetric/asymmetric features, an evolutionary algorithm can be used to evolve a set of components that are able to self-assemble into the target structure.

6.5 Summary

As discussed in *section 2.2.5: The Complexity of Self-Assembly*, designing self-assembling systems is an NP-complete problem. As a result, it will be more challenging to use traditional top-down design methodologies to create self-assembling systems as their sophistication increases. Evolutionary computing is well-suited to addressing NP-complete



3DEE1

3DEE2

3DEE3

Figure 6.11: Photographs of successful level three evolutionary experiment trials (3DEE1 - 3DEE3).

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3DPE1	Evolved	10	0	0
3DPE2	Evolved	7	3	0.002
3DPE3	Evolved	8	2	0

Table 6.9: Number of successful and unsuccessful trials for the evolutionary experiments (3DEE1 - 3DEE3) at level three, with corresponding p-values (calculated using Fisher's Exact Test (one-sided) for analyzing binary data, with respect to 0 successful and 10 unsuccessful trials for the random sets for each evolutionary experiment from there level two results)

problems. In this chapter, an extension to the three-level approach incorporating evolutionary computing was presented. An evolutionary algorithm was detailed, for evolving *good enough solutions* (a component set able to self-assemble into a single target structure). An overview of the evolutionary algorithm (including search space considerations), genotype and phenotype representations (particularly that a single genotype may have multiple phenotypes), multi-objective fitness functions (comprising a general and refined solution), and selection, crossover, and genetic operators were presented. Furthermore, one of the strengths of the evolutionary paradigm described here, is that it can be easily extended from using 2D systems to 3D systems.

The three-level approach incorporating evolutionary computing was used to conduct two 2DE and three 3DE experiments, each with a specific target structure. These experiments were conducted to test the evolutionary hypothesis of given the attributes of a closed target structure with symmetric/asymmetric features, an evolutionary algorithm can be used to evolve a set of components that are able to self-assemble into the desired target structure. The independent variable in these experiments was the set of components (evolved or random), and the dependent variable was the resulting set of self-assembled structures. The results of all the evolutionary experiments are statistically significant, with a p-value of either 0 or 0.002. The successful 2DE and 3DE experiments supports the evolutionary hypothesis. Therefore, the three-level approach incorporating evolutionary computing, and using physically encoded information, is one method to address the self-assembly design problem being NP-complete.

With the success of using the three-level approach in the context of a programming paradigm and extending it to an evolutionary paradigm, a continuation of testing the boundaries of designing physical self-assembling systems is provided in *Chapter 7: Staging the Self-Assembly Process.* Time intervals are used to divide the self-assembly process into stages to create target structures that would not otherwise be possible.

Chapter 7

Staging the Self-Assembly Process

One important challenge when creating artificial self-assembling systems is caused by the use of components that lack the plasticity of biological cells. Using components that cannot differentiate results in self-assembly being constrained to a limited set of fixed components and their bonding mechanisms (Demaine et al. 2008). One strategy to address this challenge is to divide the self-assembly process into stages, referred to as *staged* or *hierarchical* self-assembly. Demaine et al. (2008) formalized the method of staging where components can be added to, or removed from, an environment at various time intervals.

Demaine et al. (2008) demonstrated the benefits of staging theoretically using abstract tiles, where staging the self-assembly process was based on the temporal aspects of conducting laboratory experiments. In contrast, it is proposed in this chapter that one can use physically encoded information, specifically component morphology, as the dividing basis to stage the self-assembly process, inspired by biological development. This chapter considers how physical features in a set of heterogeneous, passive, mechanical components can be exploited to reduce potential assembly errors, leverage rotational bonding mechanisms, and create closed structures with symmetrical/assymerical features. This staging strategy is consistent with the definition of self-assembly (*section 1.2: Thesis Hypothesis*), as a process involving components that can be controlled through their proper design and their environment, and where components can adjust their relative positions.

Staged self-assembly provides the advantage of encoding the construction of a target structure in the staging algorithm itself and not exclusively into the design of the components. For example, a staging algorithm can be used to reintroduce previously used components and bonding mechanisms at later time intervals, prevent the formation of holes, and create more complex structure morphologies that may not be otherwise possible due to shape conflicts between components.

The sTAM extends the aTAM (*section 2.2.3: The abstract Tile Assembly Model*) by dividing the self-assembly process into time intervals. Components can be added to, or removed from, as set of environments, mirroring the laboratory operations of adding/filtering DNA-based components to solutions that can be mixed together. The sTAM has been used to investigate the algorithmic construction of structures, such as a fully connected $n \times n$ square ($n \in \mathbb{N}$). The construction of a square is problematic, as assembling tiles must be coordinated to prevent the occurrence of holes. The sTAM has shown an algorithmic efficiency with minimal tile sets and bonding mechanisms (not requiring co-operative bonding, at temperature one) in the construction of such structures (Demaine et al. 2008). This efficiency is due to staging, and is an advantage over the aTAM itself that relies on co-operative bonding, or other extensions to the aTAM that use either changes in temperature or by varying the concentration of tiles (*section 2.2.4: Extensions to the abstract Tile Assembly Model*).

Situated development is another method investigating staged construction, where artificial evolution was used to evolve the assembly plan of a structure (Reiffel & Pollack 2005). Based on rapid prototyping, assembly plans were evolved using *permanent* and *temporary* components which were "dropped" in an environment. Temporary components act as scaffolding and can be removed (representing how support material can be removed in rapid prototyping).

In contrast to Demaine et al. (2008) and Reiffel & Pollack (2005), physical examples of staged self-assembly include where staging relies on templates to enable the self-assembly process (*section 2.1: Physical Self-Assembling Systems*). Despite this work, there is little (if any) literature that describes the use of component physical information to stage the self-assembly process.

The purpose of the 2D staged (2DS) experiment and 3D staged (3DS) experiments is to demonstrate, as proof-of-concept, that staging can enable the self-assembly of closed target structures with symmetric/asymmetric features not otherwise possible. An extension to the three-level approach incorporating staging is presented next, followed by the 2DS and 3DS experiments. Research described in the staged experiments has been published in Bhalla et al. (2011b).

7.1 Staging and the Three-Level Approach

The three-level approach is extended here to incorporate a staging strategy based on exploiting component physical information. At level one, a new system rule is introduced to specify which components are present at a particular time interval. To accommodate this new rule, an extension to a self-assembly model based on the aTAM is provided at level two. Finally, physical features of components that are exploited in this staging strategy are described at level three.

7.1.1 Staging - Level One: Definition of Rule Set

At level one, the same component, environment, and systems interaction (fits and breaks) rules presented in *section 4.1: Level One: Definition of Rule Set* are used for staging. A modification to the system rule now includes specifying component type and frequency in each time interval (ψ). Time intervals indicate when components are added to a single environment (e.g. ψ_0; using a subscript 0 to n, where $n \in \mathbb{N}$ and 0 indicates the start of the self-assembly process). In the staged experiments to follow, components can only be added to the same one-pot-mixture environment.

7.1.2 Staging - Level Two: Virtual Execution of Rule Set

At level two, the 2DcTAM and the 3DcTAM (*section 4.2: Level Two: Virtual Execution of Rule Set*) is extended to incorporate staging. The extended model is referred to as the 2D and 3D staged concurrent Tile Assembly Model (2DscTAM and 3DscTAM). As with the 2DcTAM and the 3DcTAM, self-assembly is modeled in parallel, no seed tiles are required, tiles are permitted to rotate, tiles are in the in a one-pot-mixture, and the environment temperature is one.

The 2DscTAM and the 3DscTAM use a common algorithm, *staged concurrent Tile Assembly Algorithm*. Pseudocode for this algorithm is provided below. This algorithm has a loop to accommodate staging, where the algorithm followed by the 2DcTAM and the 3DcTAM, *concurrent Tile Assembly Algorithm*, is used at each stage.

```
Algorithm staged concurrent Tile Assembly Algorithm
Input: a set of staged multiset components StagedSet; 0 to $n$, where the first set indicates
    the start of the staged self-assembly process
Output: set of self-assembled target structures and remaining tiles
create an empty results set ResultSet for tiles and substructures
for $i \leftarrow 0$ to the (size of StagedSet $-$ 1)
        if $i \neq 0$
            then set all assembly locations labelled unmatchable in ResultSet to open
        add elements from StagedSet[$i$] to ResultSet
        ResultSet $\leftarrow$ concurrent Tile Assembly Algorithm(ResultSet)
    return ResultSet
```

Figure 7.1 provides an example of the 2DscTAM. As with the 2DcTAM and the 3DcTAM, a post-evaluation is conducted to test for any violations between the resulting set of self-assembled structures and the environment. Accounting for errors, including

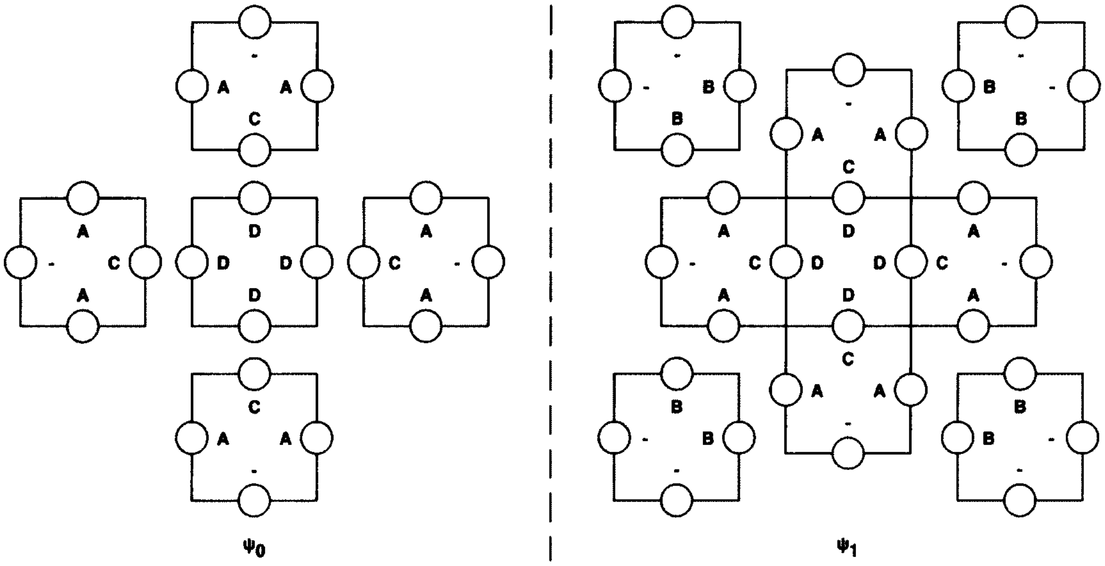


Figure 7.1: 2DscTAM example where two time intervals are used to self-assemble a 3×3 square target structure.

neutral site, uncomplimentary information, and boundary violations, and the requirement for an assembly path (Figure 4.4, page 91) are also incorporated into the 2DscTAM and the 3DscTAM.

7.1.3 Staging - Level Three: Physical Realization of Rule Set

At level three, the second 2D and 3D physical information encoding schemes are used, Table 4.5 (page 97) and Table 4.7 (page 101) respectively. Component physical features including key and lock shapes and magnetic-bit patterns – the morphologies of the components – are used to divide the self-assembly process into stages in the self-assembling systems provided in the remainder of this chapter.

Designing staged self-assembling systems using component physical features is similar to the programming paradigm, where the connectivity of a target structure was used to assign component information and determine the frequency of each component type. However in this case, component types that have the potential to lead to conflicts are separated into different time intervals. A detailed example of designing a staged self-assembling system is provided in *section 8.2: Leveraging Limited Rule Sets.* Due to the types of static self-assembling systems used in this work, a staged self-assembly process is an *additive* process where a target structure is constructed from the inside out. Detailed examples of designing staged self-assembling systems are used to achieve the goal of the staging strategy presented in this chapter, which is to demonstrate how component physical information can be used to further reduce errors (*section 7.3: 2D Staging Experiments and Results*) and leverage rotational properties (*section 7.4: 3D Staging Experiments and Results*) by using staging.

7.2 Hypothesis Statement

The hypothesis for the staged experiments was,

Hypothesis: staging based on component physical information can be used to enable components to self-assemble into closed target structures with symmetric/asymmetric features.

The three-level approach incorporating staging was used to test this hypothesis, virtually (level two) and physically (level three). One 2DS and three 3DS experiments (referred to as 2DSE1 and 3DSE1 to 3DSE3 respectively) were used to test this hypothesis. A target structure was assigned to each experiment (provided in *section 7.3: 2D Staging Experiments and Results* and *section 7.4: 3D Staging Experiments and Results*). In contrast to the programming and evolutionary experiments, enough components were supplied to create one 2D target structure and up to two 3D target structures. These target structures use more components than the previous target structures, and the num-

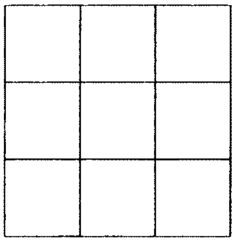


Figure 7.2: 2D staged target structure for the 2DS experiment (2DSE1).

ber of components required by these staged target structures correspond to the capacity of the environment. Ten trials were run for each experiment. A trial was evaluated to be successful if all 2D and 3D target structures were created at level two, and if at least one 2D and 3D target structure was created at level three. The independent variable in these experiments is the use of two time intervals. The dependent variable is the resulting self-assembled structures. For each experiment, a *staged* component set was specified along with a *non-staged* component set to test the independent variable.

7.3 2D Staging Experiments and Results

One 2DS experiment was conducted. Figure 7.2 provides the target structure for this experiment. The staging strategy for creating the 2D 3×3 square target structure is to construct the *centre* and *edges* of the square in the first time interval, and construct the *corners* of the square in the second time interval (Figure 7.3). In the first time interval, potential errors between the edge components can be reduced by appropriate selection of 3-magnetic-bit patterns and the use of lock shapes to assemble to the centre component. The morphology of the substructure after the first time interval has corner features that can reduce assembly errors with the use of corner components that use only lock assembly shapes. The neutral edges of the corner components effectively block a corner component from assembling to the substructure in an improper orientation (Figure 7.3).

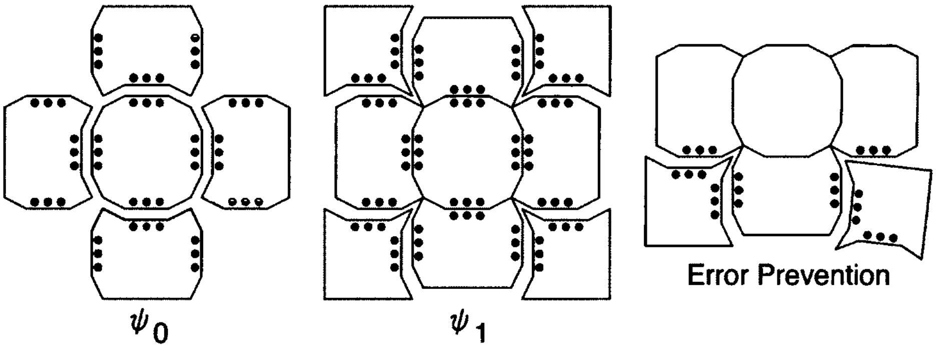


Figure 7.3: Staging strategy for the 2D target structure, and error prevention due to shape and proper 3-magnetic-bit pattern selection (e.g. avoid magnetic repulsion configuration).

The experimental procedures and results are presented according to the three-level approach: defining a staged rule set (level one), modelling using the 2DscTAM (level two), and physically testing systems using the second 2D physical information encoding scheme (level three).

7.3.1 Level One: Rule Set for 2DS Experiment

Table 7.1 provides the component rules. The control group represents components that were not divided into time intervals (non-staged). The experimental group used the same components, but divides them into two time intervals (staged). The system interaction rules from Table 4.5 (page 97) were applicable to both groups.

7.3.2 Level Two: Virtual Execution of Rule Set for 2DS Experiment

For each 2DS trial, the 2DscTAM was used to virtually evaluate the ability of each rule set (*staged* and *non-staged*) to create the 2D target structure.

2DSE1	ψ_0\{1 X (D,D,D,D), 4 X (B,-,B,C)\} ψ_1 \{4 X (-,A,A,-)\}

Table 7.1: Staged component set (represented as '*psi* # × (Top, Left, Bottom, Right)', where ψ identifies the time interval, # represents quantity, and the directions refer to component information) for the 2DS experiment (EX)

Level Two: Experimental Setup for 2DS Experiment

The component rules from Table 7.1 were mapped to an abstract representation for the 2DscTAM. The same parameter settings for the 2DcTAM from *section 5.2.2: Level Two: Experimental Setup for 2DP Experiments* were used: each component's shape was a unit square, the environment boundary size was 10×10 units, and the environment temperature was set to one. Again, a different random seed was used to initialize the 2DscTAM for each of the ten trials.

Level Two: Experimental Results for 2DS Experiment

The staged components successfully created one target structure in each of the ten trials. None of the non-staged components were able to create one target structure. The unsuccessful non-staged trials either resulted in a set of substructures (due to edge and corner components assembling in incorrect orientations), or the creation of a 3×3 open square. The results at level two were analyzed using Fisher's Exact Test (one sided) for binary data. The results are statistically significant with a p-value of 0 (Table 7.2).

7.3.3 Level Three: Physical Realization of Rule Set for 2DS Experiment

A level three translation was preformed for both the staged and non-staged components (to observe the physical results of non-staged components).

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2DSE1	Staged	10	0	0
	Non-Staged	0	10	

Table 7.2: The number of successful and unsuccessful trials for the staged and non-staged trials for 2DSE1 at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data

Level Three: Experimental Setup for 2DS Experiment

Component mapping followed Table 4.5 (page 97). The physical experimental procedure followed *section 5.2.3: Level Three: Experimental Setup for 2DP Experiments*, with the exception that two 10 minute time intervals were used for the staged component set, and one 20 minute time interval was used for the non-staged component set. Components were randomly placed initially on the surface of the tray. New components added to the second time interval of the staged trials were also randomly placed on the surface of the tray (without disturbing any components/substructures from the previous time interval). At the conclusion of each time interval, the state of the system was recorded: number of target structures, number of matching errors, and number of assembly errors. Details regarding the physical experimental setup for the 2DP experiment is provided in *section C.2: Experimental Procedure for the 2D Systems.*

Level Three: Experimental Results for 2DS Experiment

Figure 7.4 provides an example of the end of each time interval of a successful trial. Table 7.3 shows the number of successful and unsuccessful trial for the staged and non-staged trials. For both component groups, no matching and assembly errors were observed in the ten trials. Only partial structures were observed, and no open 3×3 squares, were observed at the conclusion of the non-staged trials. Using Fisher's Exact Test, this ex-

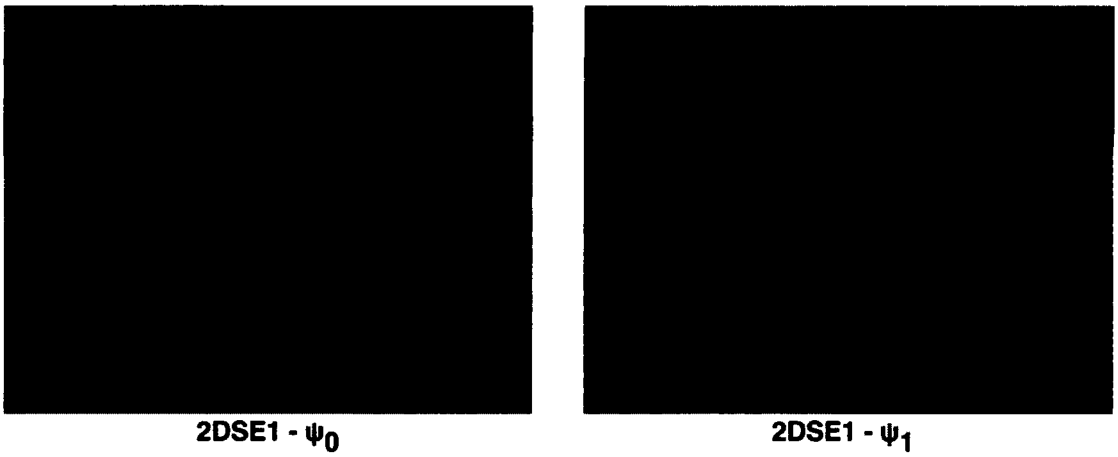


Figure 7.4: Photographs of the successful 2DS experiment (2DSE1) after the first time interval (left) and the second time interval (right).

2DSE1	Staged	7	3	0.002
	Non-Staged	0	10	

Table 7.3: The number of successful and unsuccessful trials for the staged and non-staged trials for 2DSE1 at level three, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data

periment is statistically significant at the 0.01 level (i.e. there is a 99% certainty the results are not due to chance).

7.4 3D Staging Experiments and Results

Three 3DS experiments were conducted, and a target structure was assigned to each experiment (Figure 7.5). The three 3D target structures have a three-component common *core* structure, and vary in the number of *periphery* components (increasing from two, three, and four). The core structure requires two specialized 90° bonds, whereas the

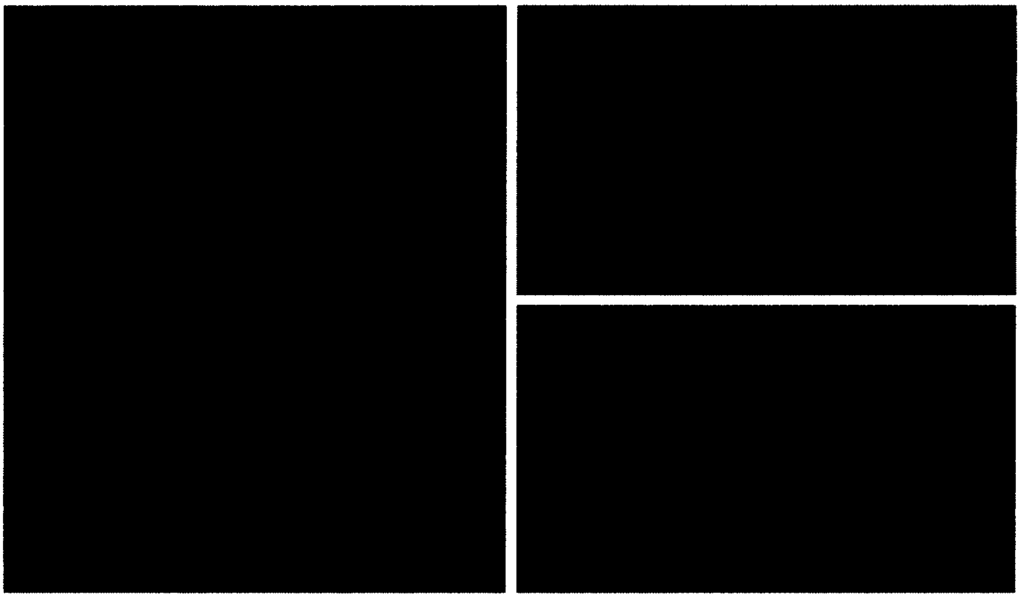


Figure 7.5: From left to right: target structures for the 3DS experiments corresponding to 3DSE1 to 3DSE3 respectively (with perspective, top, and front views).

periphery components only require general 360° bonds. As observed in the preliminary 3D experiments, substructures consisting of at least three components are not able to assemble together. Given that the likelihood of general 360° bonds occurring is more likely than specialized 90° bonds, the staging strategy for creating the three 3D target structures is to construct the core substructure in the first time interval, and construct the periphery substructures in the second time interval (Figure 7.6). The first time interval leverages the specialized component rotational information. Lock shapes for the 360° bonds are used as part of the morphology of the components in the first time interval, to reduce potential matching errors between specialized and general bonds. Furthermore, the resulting morphologies of the resulting core substructures at the end of the second time interval consist only of neutral and lock shapes, preventing assembly between the core substructures.

The experimental procedures and results are presented according to the three-level

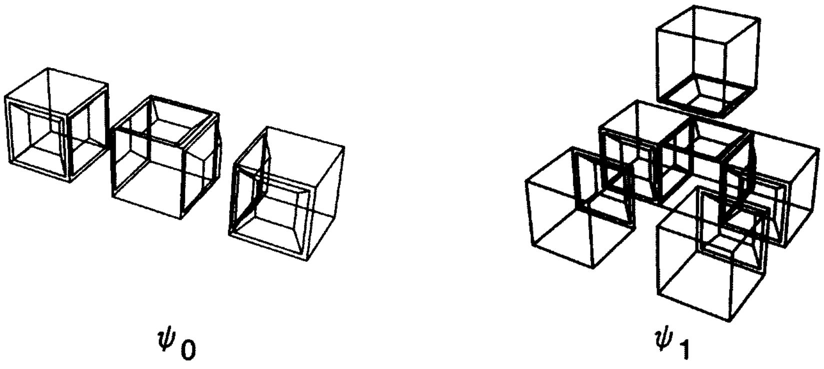


Figure 7.6: Staging strategy for the second 3D target structure (applicable to the first and third target structure).

approach: defining a staged rule set (level one), modelling using the 3DscTAM (level two), and physically testing systems using the second 2D physical information encoding scheme (level three).

7.4.1 Level One: Rule Set for 3DS Experiments

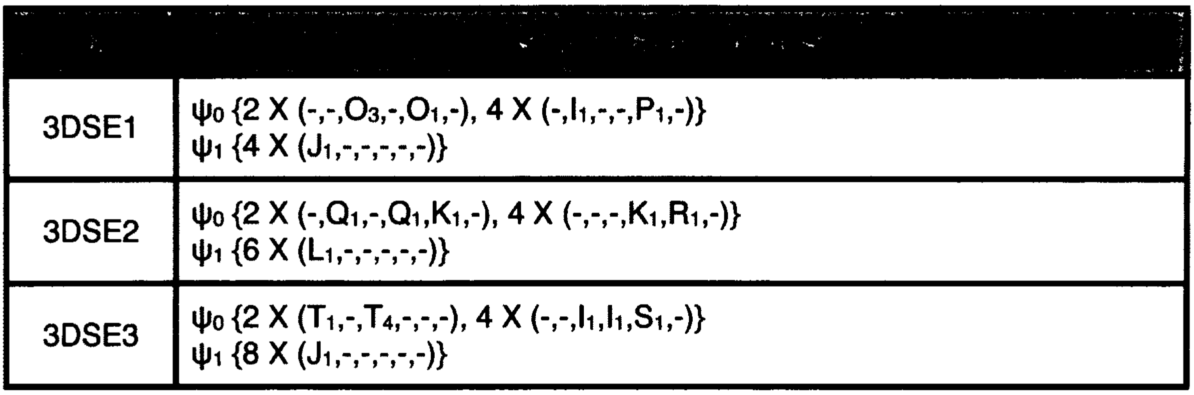
The component rules for the 3DS experiments is provided in Table 7.4. Control groups and experimental groups represent non-staged and staged (using two time intervals) component sets respectively. System interaction rules from Table 4.7 (page 101) applied to both staged and non-staged component sets.

7.4.2 Level Two: Virtual Execution of Rule Set for 3DS Experiments

For each 3DS trial, the 3DscTAM was used to virtually evaluate the ability of each rule set, staged and non-staged, to create the 3D target structures.

Level Two: Experimental Setup for 3DS Experiments

With the exception of component rules following Table 7.4, the same level two experimental setup from the 3DP experiments was used for the level two 3DS experiments



3DSE1	ψ_0 {2 X (-,-,O_3,-,O_1,-), 4 X (-,I_1,-,-,P_1,-)} ψ_1 {4 X (J_1,-,-,-,-,-)}
3DSE2	ψ_0 {2 X (-,Q_1,-,Q_1,K_1,-), 4 X (-,-,-,K_1,R_1,-)} ψ_1 {6 X (L_1,-,-,-,-,-)}
3DSE3	ψ_0 {2 X (T_1,-,T_4,-,-,-), 4 X (-,-,I_1,I_1,S_1,-)} ψ_1 {8 X (J_1,-,-,-,-,-)}

Table 7.4: Staged component set (represented as '*psi* # × (Top, Left, Bottom, Right, Front, Back)', where ψ identifies the time interval, # represents quantity, and the directions refer to component information) for the 3DS experiment (EX)

(*section 5.3.3: Level Two Experimental Setup for the 3DP Experiments*).

Level Two: Experimental Results for 3DS Experiments

The staged components, for each experiment, successfully created two target structures in each of the ten trials. Whereas, the non-staged components were not able to create a target structure. As expected, the unsuccessful non-staged components resulted in substructures consisting of three components (favouring assemblies with 360° bonds) or two components. The results at level two are statistically significant with a p-value of 0 using Fisher's Exact Test for binary data (Table 7.5).

7.4.3 Level Three: Physical Realization of Rule Set for 3DS Experiments

As with the 2DS experiment, a level-three translation was performed for both staged and non-staged components (to observe the physical results of non-staged components).

Level Three: Experimental Setup for 3DS Experiments

Component mapping followed Table 4.7 (page 101). The same level three physical experimental setup used in the 3DP experiments (*section 5.3.3: Level Three: Experimental*

3DSE1	Staged	10	0	0
	Non-Staged	0	10	
3DSE2	Staged	10	0	0
	Non-Staged	0	10	
3DSE3	Staged	10	0	0
	Non-Staged	0	10	

Table 7.5: The number of successful and unsuccessful trials for the staged and non-staged trials for 3DSE1 - 3DSE3 at level two, with corresponding p-value calculated using Fisher's Exact Test (one-sided) for analyzing binary data

Setup for 3D Experiments) was used for the level three 3DS experiments, with the exception of using two 20 minute time intervals for staged trials and one 40 minute trial for non-staged trials, and jar placements on the orbital shaker (*sectionC.3: Experimental Procedure for the 3D Systems*). At the conclusion of each time interval, the state of system was recorded (number of target structures created, matching errors, rotation errors, and assembly errors).

Level Three: Experimental Results for 3DS Experiments

The 3D level-three results are provided in Figure 7.7 (page 163), along with examples of the end of each time interval of a successful staged trial. For each experiment, no matching and assembly errors were observed in the ten trials. Rotational errors were observed in each staged experiment (Figure 7.8, page 164), and each non-staged experiment (Figure 7.9, page 165). Using Fisher's Exact Test, the first two 3D experiments are statistically significant at the 0.05 level and the third experiment is statistically significant at the 0.50 level (i.e. there is a 95% and 50% certainty the results are not due to chance).

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3DSE1	Staged	4	6	0.043
	Non-Staged	0	10	
3DSE2	Staged	5	5	0.016
	Non-Staged	0	10	
3DSE3	Staged	1	9	0.500
	Non-Staged	0	10	

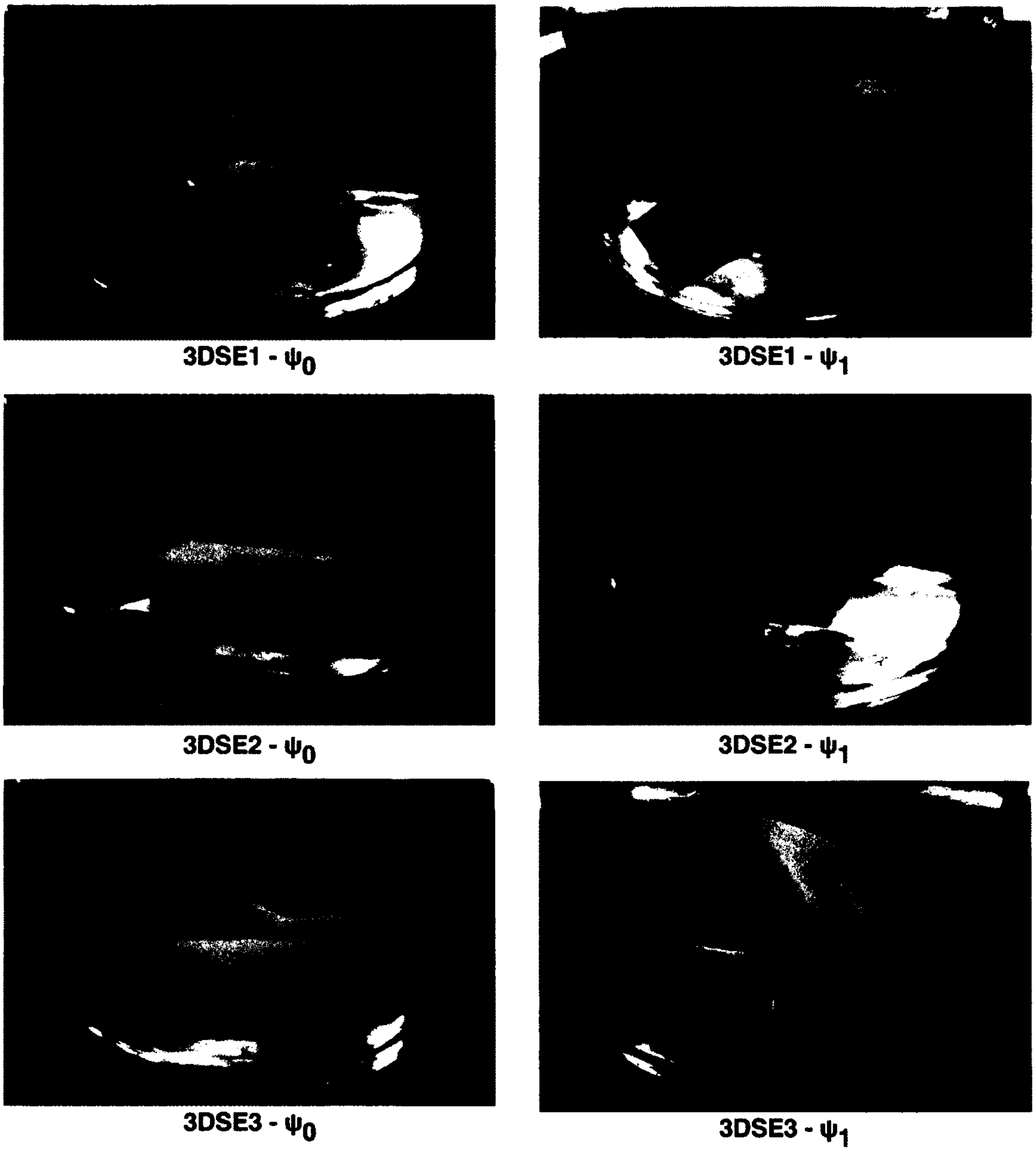
Table 7.6: Number of successful and unsuccessful trials for the staging experiments (3DSE1 - 3DSE3) at level three, with corresponding p-values calculated using Fisher's Exact Test (one-sided) for analyzing binary data

Even though one successful staged trial was observed with the third 3D experiment, we do not consider the result statistically relevant.

7.5 Summary

Staging is an essential part of biological development. Staged self-assembly provides the advantage of encoding the construction of a target structure in the staging algorithm itself and not exclusively into the design of the components. In contrast to other staging methods, based on the temporal aspects of conducting laboratory experiments or relying on templates to enable the self-assembly process, the staging strategy presented in this chapter uses physically encoded information as the dividing basis.

An extension to the three-level approach incorporating staging was presented. A new system rule specifying component type and frequency for each stage was given at level one. Two new analytical tools were detailed at level two. The 2DscTAM and the 3DscTAM use staged, parallel self-assembly, do not require seed tiles, permit component



3DSE1 - ψ_0

3DSE1 - ψ_1

3DSE2 - ψ_0

3DSE2 - ψ_1

3DSE3 - ψ_0

3DSE3 - ψ_1

Figure 7.7: Photographs of successful level three staged experiment trials (3DSE1 - 3DSE3) after the first time interval (left) and the second time interval (right).

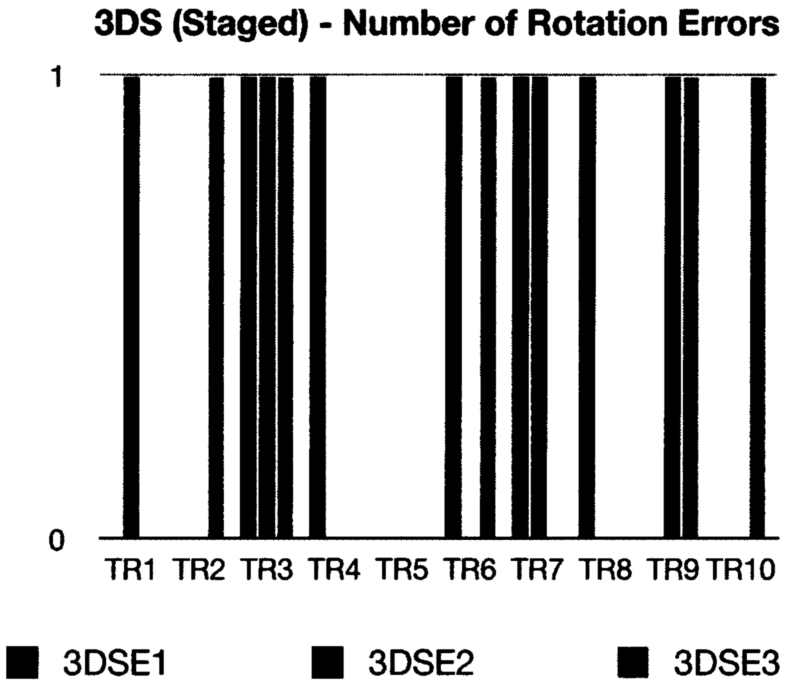


Figure 7.8: Rotational errors at the end of each 3DS staged trial.

rotation, at temperature one. Furthermore, the 2DscTAM and the 3DscTAM are based on the 2DcTAM and the 3DcTAM and are likewise Turing universal. At level three, it was proposed that the interplay between component physical information (shape and magnetic patterns) could be used to reduce assembly errors and leverage rotational properties by using staging.

Four proof-of-concept experiments were provided. The independent variable in these experiments was the use of two time intervals. The dependent variable was the resulting set of self-assembled structures. All the staged component sets, except for the 3DSE3 experiment, were able to successfully construct their respective target structures at a statistically significant level (with 99% and 95% confidence for the 2DS and 3DS experiments), at level three. The successful results provide evidence to support the hypothesis, that staging can be used to enable components to self-assemble into closed target struc-

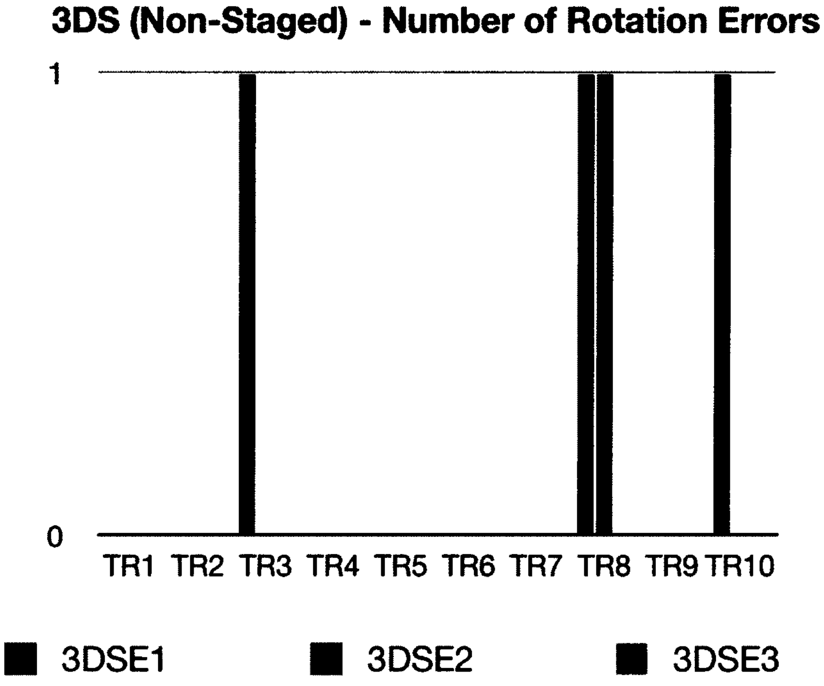


Figure 7.9: Rotational errors at the end of each 3DS non-staged trial.

tures with symmetric/asymmetric features.

An implication of staging is on the self-repairing properties of a system. Although it was observed that the 2D 3×3 square was able to self-repair, this was only within the second stage. Further research into features that allow for, and the understanding of the limits to, self-repair between specific stages is required to continue to further develop this staging paradigm. For example, although salamanders undergo development through unique stages, they can regrow lost limbs by repeating earlier developmental stages (Wolpert 1998).

Chapter 8

Discussion

In this chapter, the 2D and 3D experimental results from *Chapter 5: Programming Self-Assembling Systems*, *Chapter 6: Evolving Self-Assembly Rule Sets*, and *Chapter 7: Staging the Self-Assembly Process* are summarized and analyzed. Next an extension to the staging paradigm is presented. Future work is examined, including an additional physical information encoding scheme, and considering error correction and scaling methods. Finally, applications of the research presented in this thesis are discussed.

8.1 Summary and Analysis of the Experimental Results

The self-assembly design methodology, the three-level approach (presented in *Chapter 4: The Three-Level Approach to Self-Assembly Design*) was developed to test the thesis hypothesis (*section 1.2: Thesis Hypothesis*) and determine the feasibility of encoding information as physical components and their corresponding environments to enable the self-assembly of closed structures with desired morphologies. The three phases of the three-level approach include: (1) definition of rule set, (2) virtual execution of rule set, and (3) physical realization of rule set. The motivation behind the three-level approach is in finding the fundamental information structures and rules in theory (level one), testing and refining those rules through simulation (level two), and testing and refining those rules through embodied physical experiments (level three).

Proof-of-concept experiments were presented, where different design paradigms used the three level-aproach, including: programming (*Chapter 5*), evolving (*Chapter 6*), and staging (*Chapter 7*). Experiments were conducted using 2D and 3D systems for each paradigm (programming: 2DPE1 - 2DPE3, and 3DPE1 - 3DPE5; evolving: 2DEE1 and 2DEE2, and 3DEE1 - 3DEE3; and staging: 2DSE1, and 3DSE1 - 3DSE3). Each experiment was assigned a unique closed target structure with symmetric/asymmetric features. Due to the morphology of each target structure, the underlying lattice pattern of the components (square and cubic lattices) was insufficient for components to self-assemble into the target structures in their corresponding environments. Therefore, it was appropriate to determine if the information encoded in a set of components was sufficient to achieve the applicable target structure by self-assembly in their environment, for each experiment.

With the exception of 2DSE1, the creation of multiple target structures in parallel was considered. Ten trials were conducted for each experiment. A trial at level two was considered successful if all three of the applicable target structures were achieved. A trial at level three was considered successful if at least one target structure was achieved. The difference in the number of target structures for successful evaluation was due to physical self-assembly being more challenging than modelling. In the programming and evolutionary experiments, the independent variable was a set of components, and the dependent variable was the resulting self-assembled structures. In contrast to the staging experiments, the independent variable was the use of two time intervals, and the dependent variable was again the resulting set of self-assembled structures. As a result, two groups of components were specified for each experiment, one designed using the three-level approach and one as a control group. The control groups for the programming and evolutionary experiments were randomly generated. The control group for the staging experiments used only a single time interval. This method of creating the designed and control component sets was chosen as part of the experimental setup with the aim of providing proof-of-concept evidence to test the thesis hypothesis, by conducting a statistical significance test. Fisher's Exact Test (one-sided) for analyzing binary data was selected to complement the successful or unsuccessful evaluation of each trial, for each experiment.

Table 8.1 summarizes the results from the level two experiments presented in this thesis. Each experiment was successful, with a p-value of 0. The successful results validate the use of the self-assembly models developed, the 2DcTAM/3DcTAM (*section 4.2: Level Two: Virtual Execution of Rule Set*) and the 2DscTAM/3DscTAM (*section 7.1.2: Staging - Level Two: Virtual Execution of Rule Set*). The level two results justify the benefits of the unique features of these models, including:

- modelling self-assembly concurrently,
- not requiring seed tiles (components),
- permitting tile rotations,
- accommodating 2D and 3D tiles, and
- using an environment temperature of one.

Furthermore, these successful level two results support the three different paradigms (programming, evolving, and staging) using the three-level approach in their ability to be used to design self-assembling systems.

There were two reasons for the unsuccessful control group results. The first reason was the creation of inappropriate substructures. These substructures would either be open but prohibit the creation of a target structure (due to the arrangement of the components), be closed and therefore not allow for the possibility of creating a target structure, or create undesirable features such as holes (i.e. for the 3×3 square target structure for 2DSE1). The second reason, which is only applicable to the random component sets, is that some sets had components that did not have complementary information to the

2DP	2DPE1	10	0	0
	2DPE2	10	0	0
	2DPE3	10	0	0
2DE	2DEE1	10	0	0
	2DEE2	10	0	0
2DS	2DSE1	10	0	0
3DP	3DPE1	10	0	0
	3DPE2	10	0	0
	3DPE3	10	0	0
	3DPE4	10	0	0
	3DPE5	10	0	0
3DE	3DEE1	10	0	0
	3DEE2	10	0	0
	3DEE3	10	0	0
3DS	3DSE1	10	0	0
	3DSE2	10	0	0
	3DSE3	10	0	0

Table 8.1: Summary of the level two experiments, showing the number of successful and unsuccessful trials for the designed component groups of the programming (2DP and 3DP), evolving (2DE and 3DE), and staging (2DS and 3DS) experiments, and corresponding p-values calculated using Fisher's Exact Test (one-sided) for analyzing binary data where all the control component groups had zero successful and ten unsuccessful trials

other components in the set. As a result, these components could not be included in the self-assembly process.

With the success of all the level two experiments, a level three translation was conducted for the programmed and evolved (designed) component sets (and not their corresponding random component sets as they were unsuccessful), and for both groups for the staged experiments (as both groups shared the same components, testing the non-staged component sets was conducted to observe the physical results and was feasible since there was no additional financial cost to construct components and run experiments). Table 8.2 summarizes the results of the level three experiments presented in this thesis. P-values were calculated for the programming and evolutionary experiments, with respect to the zero successful and ten unsuccessful trials for the random component sets from the level two results. P-values for the staging experiments were calculated using the level three results of both the staged and non-staged component sets. The programming and evolutionary physical experiments are statistically significant at the 0.01 level, and with the exception of 3DSE3, the staged physical experiments are statistically significant at the 0.05 level. There is a 99% and 95% certainty these results are not due to chance. Therefore, these physical results are considered to be statistically relevant.

There was one successful trial in 3DSE3. This experiment is statistically significant at the 0.50 level (i.e. there is a 50% certainty the results are not due to chance). Despite one successful trial, this experiment is not considered to be statistically relevant. In general, the results from all the staged experiments are lower than those achieved by all the programmed and evolved experiments. This is consistent with the outlook that the staged experiments would be more challenging, since they use more components with specialized 90° rotational information in comparison to the programmed and evolved experiments.

At the end of each trial of the 2D experiments, the state of the system was recorded,

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2DP	2DPE1	10	0	0
	2DPE2	9	0	0
	2DPE3	7	3	0.002
2DE	2DEE1	10	0	0
	2DEE2	9	1	0
2DS	2DSE1	7	3	0.002
3DP	3DPE1	10	0	0
	3DPE2	7	3	0.002
	3DPE3	9	1	0
	3DPE4	7	3	0.002
	3DPE5	9	1	0
3DE	3DEE1	10	0	0
	3DEE2	7	3	0.002
	3DEE3	8	2	0
3DS	3DSE1	4	6	0.043
	3DSE2	5	5	0.016
	3DSE3	1	9	0.5

Table 8.2: Summary of the level three experiments, showing the number of successful and unsuccessful trials for the designed component groups of the programming (2DP and 3DP), evolving (2DE and 3DE), and staging (2DS and 3DS) experiments, and corresponding p-values calculated using Fisher's Exact Test (one-sided) for analyzing binary data where all the control component groups had zero successful and ten unsuccessful trials

including: number of target structures, number of matching errors (between conflicting physical information, where no fits rule is applicable), and number of assembly errors (partial attachment between complementary physical information). The components in 2DPE1 created the largest number of target structures, as this experiment used a combination of the fewest components and error-free 3-magnetic-bit patterns in relation to their environment size and temperature, in comparison to the other 2D experiments. No matching errors were observed in the 2D experiments. Assembly errors were only observed in 2DPE2 and 2DPE3. In addition to these quantitative observations, qualitative observations were also made. In the programmed and evolved 2D experiments, it was observed that the emerging substructures would at times create a spatial relationship between them that would negatively impact the self-assembly process. However, it was also observed in the 2D programmed and evolved experiments that self-repair was possible. Self-repair was also observed in the second time interval of 2DSE1. Lastly, it was difficult to implement a physical environment temperature of one. A number of factors, including the number of components and their physical information, the size and shaking-level of the environment, and the emerging substructures all play a role in the environment temperature of these mechanical systems.

Likewise for the 3D experiments, quantitative measures with the addition of rotational errors (between complementary 3D physical information) were recorded at the end of each trial. Experiments using the fewest number of components and error-free 5-magnetic-bit patterns produced the largest number of target structures. As with the 2D experiments, no matching errors were observed in the 3D experiments. In contrast to the 2D experiments, no assembly errors were observed in the 3D experiments. However, rotational errors were observed in the 3D programmed and staged experiments. Additional qualitative observations, such as self-repair and the interactions between emerging substructures, were not possible due to the experimental setup of the 3D experiments.

However, it was again difficult to implement a physical environment temperature of one. It is believed that this discrepancy in achieving an accurate environment temperature is one of the reasons for observing rotation errors.

Both the 2D and 3D experiments demonstrate the benefits of permitting component rotations. Consequently, permitting rotations allows for multiples of the same component type to be used in a system, and allows for the exploitation of symmetry in a target structure (*section 2.2.5: The Complexity of Self-Assembly*). The emergence of target structures with asymmetric features in the 2D and 3D experiments demonstrate how the use of physically encoded information in a component set or the use of specialized rotational information in 3D systems can be used to achieve such features (*section 2.2.1: Physical Information Encoding in Nature*).

The design of successful 2D and 3D experiments was due to the three-level approach, which is inherently bottom-up by being able to map a set of rules to a physical system using physical encoded information. The three-level approach was used in the context of three design paradigms, programming, evolving, and staging. The programming experiments used the three-level approach directly and showed that all the second 2D (Table 4.5 page 97) and 3D (Table 4.7 page 101) physical information encoding schemes were capable of being used to create closed target structures with symmetric/asymmetric features. The three-level approach was extended to incorporate evolutionary computing for the evolutionary experiments. These experiments presented how component rule sets could be evolved to self-assemble into desired target structures. Furthermore, the evolutionary experiments provided a design solution to self-assembly being an algorithmically NP-complete design problem. The three-level approach was extended again using staging and demonstrated how additional information (the use of time intervals) could be used to achieve more complex target structures that would not otherwise be possible. As well, the successful 2D and 3D results demonstrate how component physical information (and in the context of staging) can be used to reduce or prevent errors from occurring during the self-assembly process, and how 3D component rotational information can be leveraged. Therefore, the successful physical experimental results at level three provide evidence to support the thesis hypothesis that it is possible to encoded information as physical components and their corresponding environment to enable the self-assembly of closed structures with desired morphologies.

8.2 Leveraging Limited Rule Sets

To expand upon the proof-of-concept experimental results presented in this thesis, particularly the 3D staging concept based on component physical information, a method to leverage a limited set of component rules using staging is discussed. As it is difficult to construct components (Demaine et al. 2008), it is important to be able to leverage a limited set of components and their corresponding rule sets. The following example was created using a target structure with the shape of a dog (Figure 8.1). This example is used to demonstrate how staged self-assembly can also be used to leverage a limited set of component interaction rules (3D 5-magnetic-bit patterns, Figure 4.8, page 98), and create body plan designs inspired by development in biological systems (Wolpert 1998).

In this example a fully-connected dog structure is desired, meaning that neighbouring components must be assembled to one another (e.g. no neighbouring neutral shapes or neighbouring neutral and lock shapes). Although fully-connected self-assembled structures are not required in general, this constraint is used here for structural rigidity concerns. Another constraint placed on this example, similar to the staged experiments presented (*Chapter 7: Staging the Self-Assembly Process*), is that each time interval follows a *one-component-step* after the initial stage. This means that the existing substructure can only grow by at most one component in all axes in Cartesian space. The staged self-

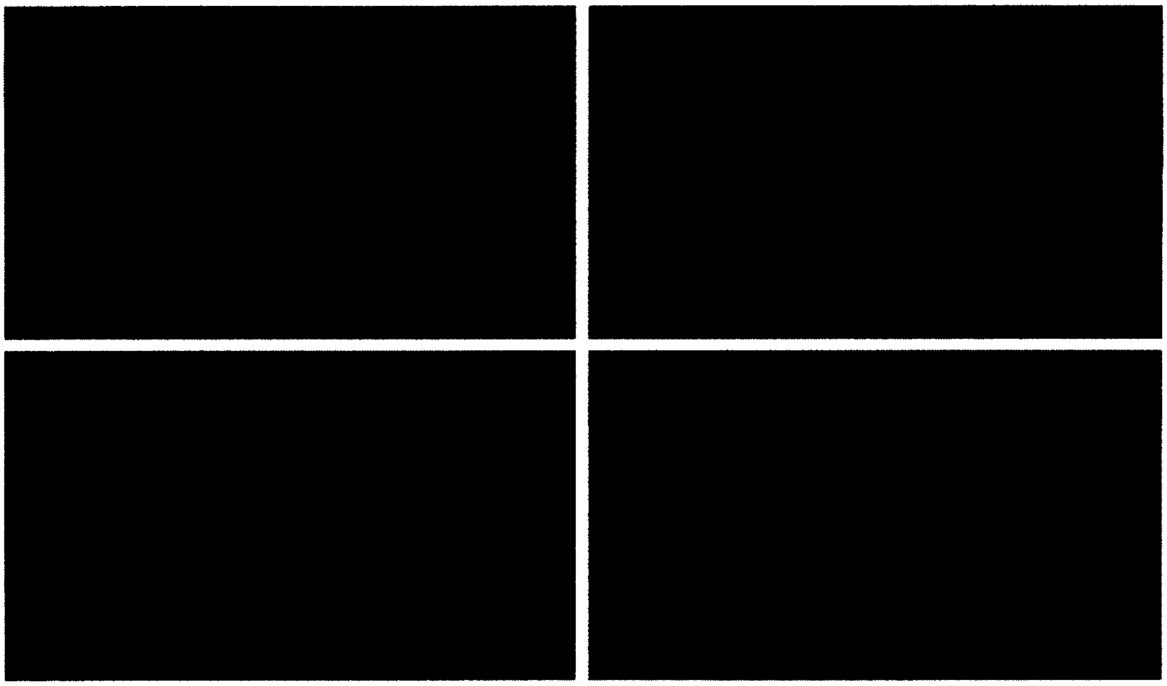


Figure 8.1: Four views of the dog target structure.

assembly of a single dog structure is presented first, followed by design considerations for the staged self-assembly of multiple dog structures in parallel.

Component types and their rotational properties is based on the connectivity of the components in the dog target structure. As shown in Figure 8.2 (page 177), the dog target structure requires six pairs of 90° rotational codes. However, only three pairs of 90° rotational codes are present in the 5-magnetic-bit pattern (Figure 4.8 page, 98). To overcome this deficiency in the number of 90° rotational codes, staging can be used to:

- reintroduce previously used component information al later time intervals,
- use the morphology of a substructure of the dog target structure (at an intermediate stage during the self-assembly process) of neighbouring components to emulate a 90° rotational code using the 180° rotational codes, and
- reduce the number of 90° rotational codes required when constructing only one

single target structure in contrast to multiple target structures, since the orientation of the overall dog target structure during its construction is not important (i.e. head versus tail facing forward).

The dog target structure can use all three types of rotational information in a single structure. Table 8.3 (page 178) lists the 5-magnetic-bit patterns assigned to key and lock shapes that are better suited to create the dog target structure. In this designation, the worst possible mismatch error between conflicting component information is a 3-out-of-5 positional match, which is the same worst-case scenario between complementary 90° rotational codes.

Using this new 5-magnetic-bit encoding scheme arrangement, a single dog structure can be made using six time intervals (Figure 8.3, page 179). In this case, time interval ψ_1 is used to determine the orientation of the head and tail. The second and third time intervals are used to create the main body, and determine the orientation of the head, neck, and direction of the head. The fourth and fifth intervals are used to build the legs and neck. The sixth interval is used to build the head, feet, and tail. This staged process takes advantage of using similar component types in the same time interval, and shows the benefits of symmetry within a time interval for parallel self-assembly. Furthermore, rotational information Q used in time interval ψ_0 is reused in time interval ψ_3 (Figure 8.3, page 179).

The disadvantage with the previous staged self-assembly process is it cannot be used to create multiple dog target structures, as it would then be possible to create dogs with either two heads or two tails in time interval ψ_1. To solve this problem, at least one pair from each of the three 90° rotational codes is required in time interval ψ_1 (Figure 8.4, page 181). However, this results in an insufficient number of pairs of 90° rotational codes to create the corner parts of the main body of the dog target structure. The 180° rotational code pair can be used due to the spatial relationship of the neighbouring

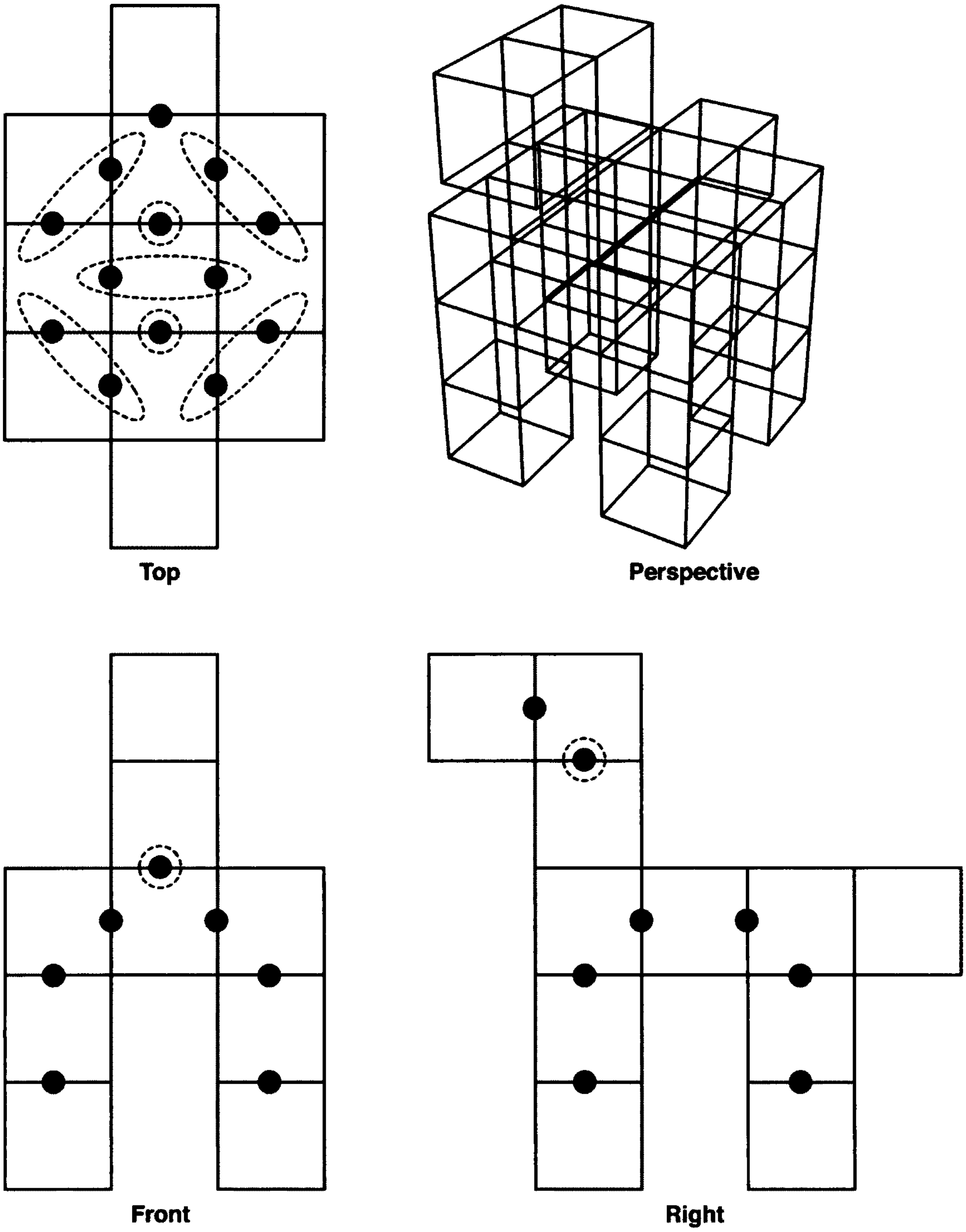


Figure 8.2: The dog target structure showing an assembly between neighbouring components (solid black circle) and the 90° rotational requirements (dashed lines), where the grouped connections can use the same rotational information due to symmetry (and the four corner components in the body can use the same rotational information, top view).

Lock	00000	I	I fits$_{360}$ I → I+J	ϕ_2 breaks I+J → I ; J
Lock	**01111**	**L**	L fits$_{360}$ K → L+K	ϕ_2 breaks L+K → L ; K
Lock	01010	M	M fits$_{180}$ N → M+N	ϕ_2 breaks M+N → M ; N
Lock	**01100**	**O**	O fits$_{90}$ P → O+P	ϕ_2 breaks O+P → O ; P
Lock	**11000**	**Q**	Q fits$_{90}$ R → Q+R	ϕ_2 breaks Q+R → Q ; R
Lock	10111	T	T fits$_{90}$ S → T+S	ϕ_2 breaks T+S → T ; S
Key	11111	J	J fits$_{360}$ I → I+J	ϕ_2 breaks J+I → J ; I
Key	**10000**	**K**	K fits$_{360}$ L → K+L	ϕ_2 breaks K+L → K ; L
Key	10101	N	N fits$_{180}$ M → N+M	ϕ_2 breaks N+M → N ; M
Key	**10011**	**P**	P fits$_{90}$ O → P+O	ϕ_2 breaks P+O → P ; O
Key	**00111**	**R**	R fits$_{90}$ Q → R+Q	ϕ_2 breaks R+Q → R ; Q
Key	01000	S	S fits$_{90}$ T → S+T	ϕ_2 breaks S+T → S ; T

Table 8.3: 5-magnetic-bit encoding scheme to create the dog target structure, where the differences between the encoding scheme used for the previous 3D experiments (Table 4.7 page 100) are highlighted

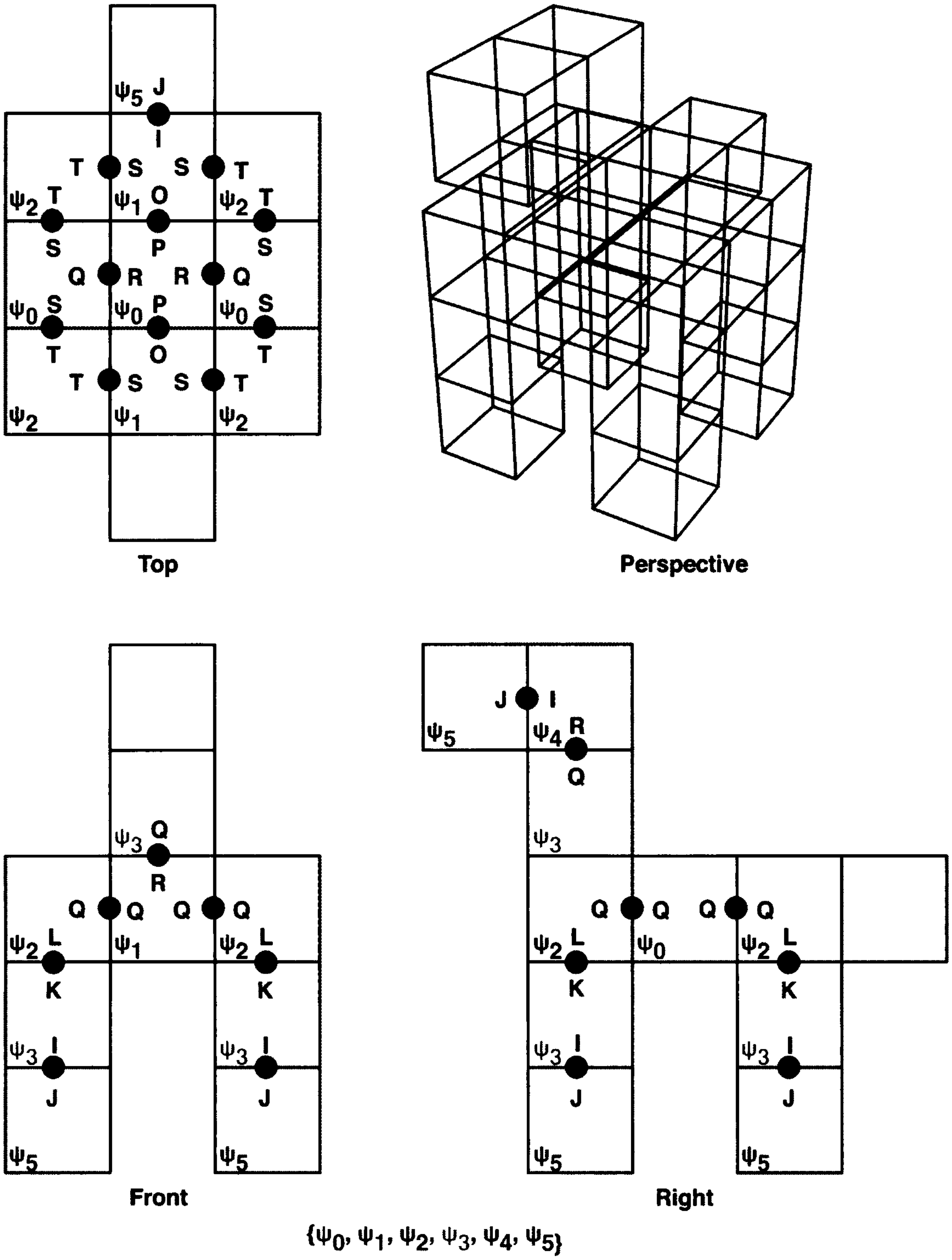


Figure 8.3: Staging for a single dog target structure, where colours are used to identify the six time intervals.

components to the corner components of the main body structure of the dog target structure (Figure 8.4). As with the staged self-assembly process to create a single dog target structure, the staged self-assembly process to create multiple dog target structures also uses six time intervals and reuses rotational information Q in time interval ψ_0 again in time interval ψ_3 (Figure 8.4).

This example shows how staged development of a body plan can be used to leverage a limited set of rules. In particular, component interaction rules can be exploited by reintroducing previously used component information at later time intervals. The physical construction of this staged dog target structure is one area of future work.

8.3 Future Work

As future work, alternative component information encoding schemes using magnetic-bit patterns is presented as a method to reduce component rotational errors in 3D systems. In addition, more broad directions of future work in self-assembly are considered.

8.3.1 Alternative Physical Information Encoding Schemes

Alternative physical information encoding schemes, particularly the 3-magnetic-bit and 5-magnetic-bit patterns presented in this thesis, should be considered to potentially reduce or prevent errors during the self-assembly process and as one avenue to create more complex self-assembling systems. As an example, a 9-magnetic-bit pattern could be used alternatively to a 5-magnetic-bit pattern with 3D components. The 3D components presented in this thesis have the capacity to accommodate nine locations (3×3) for permanent magnets in key and lock shapes.

In the 5-magnetic-bit patterns, specialized 90° rotational information had the property that self-errors were possible, resulting in a 3-out-of-5 match between complementary magnetic polarities. Although 4-out-of-5 matches could be avoided by manipulating

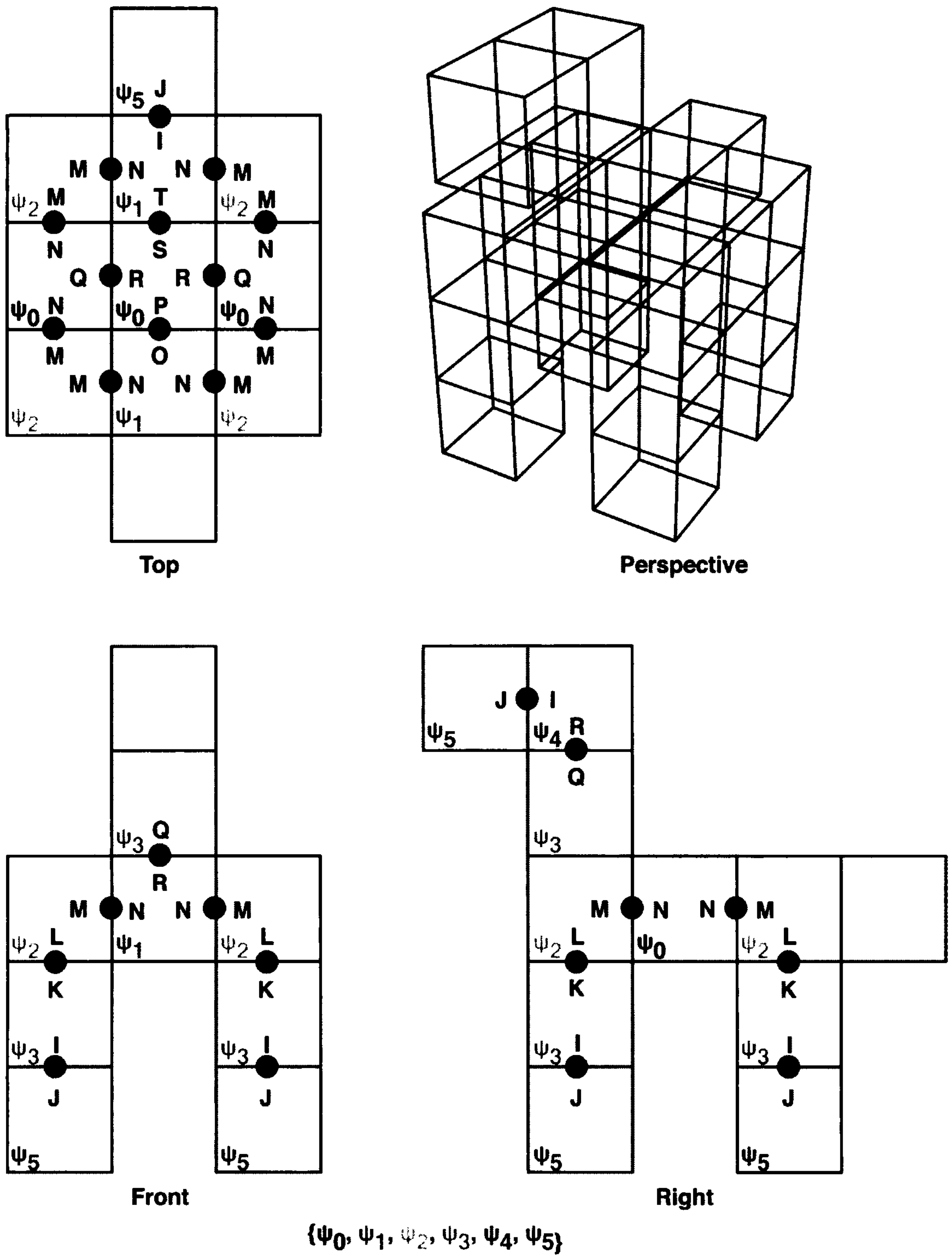


Figure 8.4: Staging for multiple dog target structures, where colours are used to identify the six time intervals (the last three time intervals correspond to the last three time intervals used to create a single dog target structure Figure 8.3 page 179).

the assignment of the 5-magnetic-bit patterns to keys and locks, and selection of which component physical information was present in a system (and each stage). This small gap between a perfect 5-out-of-5 match and an error of a 3-out-of-5 match was one of the reason for the difficulty in implementing an appropriate environment temperature.

With a 9-magnetic-bit pattern, there are seventy pairs of complementary codes. Table 8.4 shows the complementary codes and their self-matches, considering planar rotation of a component's face. Although there are several categories of information that have a high number of self-errors, these 9-magnetic-bit patterns do not need to be used. The remaining information categories have at worst a 5-out-of-9 match (highlighted in yellow, Table 8.4). This wider gap between a perfect match and self-error potentially allows for an easier implementation of an appropriate physical environment temperature to reduce or prevent errors during the self-assembly process.

Additional future work concerns regarding physical information encoding in static self-assembling systems includes the use of layered magnetic-bit patterns in 2D systems, as a way to manage component size (similar to the layered electromagnets used in Claytronics, *section 2.1.1: Robotic Systems*). As well, the use of *don't-care bits* would create one-to-many component interactions (e.g. *U fits V* → *U+V*, and *U fits W* → *U+W*). The benefit of one-to-may component interactions is that it offers another avenue to study symmetry and asymmetry in self-assembling systems.

Additional materials should also be investigated in creating more complex forms of physical information (e.g. including capillary and hydrophobic/hydrophilic interactions along with magnetic interactions at the macro and mesoscale, or Watson-Crick complementarity with magnetic interactions at the nanoscale). Lastly, future physical information encoding schemes should also be considered with more complex environment interactions, as a method to further reduce or prevent errors, and reduce the time required to complete the self-assembly process.

360	**0**	**0**	**0**	**0**	**4**
180	**0**	**8**	**0**	**8**	**2**
180	0	4	0	4	4
90	**0**	**6**	**4**	**6**	**4**
90	0	4	8	4	4
90	0	4	6	4	16
90	0	4	4	4	16
90	0	6	2	6	8
90	0	2	4	2	4
90	0	2	2	2	8

Table 8.4: The number of self-matching errors using a 9-magnetic-bit pattern and the corresponding number of pairs for each category of rotational information, where ROT-1 - ROT-4 correspond to a 90° planar rotation between the faces of two components with the same rotational information

8.3.2 Broad Directions

Along with the above future considerations regarding leveraging limited rule sets and alternative physical information encoding schemes, more broad directions of future work are given. These directions are based on the three design paradigms (programming, evolving, and staging) based on the three-level approach, the computational aspects of self-assembly, error correction methods, and scaling self-assembling systems.

Programming Self-Assembling Systems

The programming methodology used the three-level approach, and all three levels can be extended. At level one, additional rule types should be investigated. New forms of components and environments should be considered. Along with one-to-many component interactions, such as selective disassembly, selective partial assembly and catalysts should lead to more complex interactions. These types of rules could help bridge the knowledge gap between static and dynamic self-assembling systems (Whitesides & Grzybowski 2002). These additional interaction rules should also lead to self-repair, self-disassembly, and self-reconfiguration along with self-assembly rule sets. These additional rules would allow for a deeper investigation into the relationship between reductionism, emergence, and complexity under the guise of self-assembly (Whitesides & Grzybowski 2002).

An overall improvement to the three-level approach would be augmentations of the level two modelling. Such an augmentation includes the use of both an abstract, tile-based model (e.g. 2DcTAM/3DcTAM and 2DscTAm/3DscTAM) for computationally efficient evaluation of a set of self-assembly rules, along with a physics-based model for evaluation of a translated set of self-assembly rules. For example, only self-assembly rule sets that are successful in the tile-based model would be evaluated using the physics-based model for a more efficient use of computational resources. Such an augmentation would be better at detecting physically infeasible rule sets.

Along with the above considerations of alternative forms of physical information encoding, both further discrete design spaces and continuous design spaces (e.g. Bhalla & Bentley 2006) should also be considered at level three to create more complex self-assembling systems. It is anticipated that using neutrally buoyant components would avoid the layering effect seen the in the 3D systems presented. However, mixed component systems should continue to be improved, as the plethora of self-assembly examples using mixed component systems throughout nature (e.g. biological systems) provide enough motivation for their study. Although it was qualitatively observed that structures would assemble, disassemble, and new structures would emergence periodically, a better experimental setup, particularly for 3D systems, to record quantitative environment observations would assist in making improvements to the environment. New experiments for testing the various environment variables (e.g. shaking speed and duration) would also lead to more robust physical results. Evaluating the resulting environment data would be required to also consider future experiments where changes in the environment (such as fluctuations in temperature) could be used to enable the self-assembly process along with component information.

Evolving Self-Assembling Systems

Improvements to the evolutionary algorithm include conducting experiments to determine if there are better parameter settings for the number of generations, population size, and number of genotype evaluations, as well as for the selection, crossover, and genetic operators. Similarly, investigations into objective weightings, and new objectives (e.g. when considering refined solutions), should also be conducted to improve multi-objective fitness evaluations. Enhancements to the canonical genetic algorithm used here should also be considered, as another method to improve the evolved results. As future work, evolving multiple target structures should also be investigated. Lastly, evolutionary computing could also be used to evolve component and environment specifications, along with component-to-component and component-to-environment interactions.

Staging Self-Assembling Systems

Additional directions of future work based on staged self-assemble include investigating additional examples which leverage limited rule sets, studying self-repair properties and constraints between stages, and considering how and which stages can be *compressed* such that each stage does not necessarily have to follow a one-component-step assembly sequence.

Computational Aspects of Self-Assembling Systems

The aTAM links computation with self-assembly as a model of pseudo-crystalline growth. However, the resulting self-assembled structures are static. A shift towards using new rule sets that are capable of Turing universal computation for the creation of dynamic structures should also be considered. The advent of a Turing universal dynamic self-assembling systems could lead to the creation of a self-assembling computer which is able to interact with its environment. Evolving the morphologies of new active components and their environments could give rise to such self-assembling systems.

8.3.3 Error Correction

Error correction is an essential part of working towards creating more complex self-assembled structures. The aTAM was used to demonstrate how co-operative bonding can be used to reduce/prevent errors from occurring during the self-assembly process at temperature two (*section 2.2.3: The abstract Tile Assembly Model*). Templates in the environment have also been used reduce/prevent errors from occurring in physical systems (*section 2.1.2: Mesoscale Self-Assembly*). In this thesis, two additional methods to reduce/prevent errors was presented: (1) using component physical properties based on the combination of a shape space and an assembly protocol space (*section 4.3: Level Three: Physical Realization of Rule Set*), and (2) using staging based on the morphology of substructures between time intervals (*Chapter 7: Staging the Self-Assembly Process* and *section 8.2: Leveraging Limited Rule Sets*). Future methods to correct erroneous structures could include: using specialized structures to adjust misconstructed structures (as an additional form of templating), removing misconstructed structures (e.g. realized by changes in the environment), and using components that can adjust their morphologies to start, stop, and restart the self-assembly process.

8.3.4 Scaling

There are a variety of *scaling* issues in self-assembling system, including the physical size of components, designing components, and the number of components in a system. *Section 2.1: Physical Self-Assembling Systems* presented physical self-assembling systems from the macroscale, across the mesoscale, and down to the nanoscale. Fractals can be used to create structures that are self-similar at all level of magnification (*section 2.2.5: The Complexity of Self-Assembly*). Designing self-assembling systems is an algorithmically NP-complete problem (*section 2.2.5: The Complexity of Self-Assembly*). The time needed to solve an NP-complete problem using currently known algorithms increase dramatically as the size of the problem grows, e.g. the number of components. In this thesis, evolutionary computing was used to address the algorithmic constraints of designing self-assembling systems (*Chapter 6: Evolving Self-Assembly Rule Sets*). As well, this thesis provided an example of how staging can be used to reuse component information in previous time intervals to create structures with larger numbers of components (*section 8.2: Leveraging Limited Rule Sets*). The staged self-assembling systems provided require additional components to be inserted into the system. As future work, automated staging should be considered, and could be realized using partial assembly rules for example. As well, *shepherd* components could be used to guide components, and *catalytic* components could be used to amplify the creation of substructures in systems with large environments to accommodate copious components. Dynamic self-assembly could also be used to demonstrate the recycling of components, by using selective assembly and disassembly interactions, in a target structure to perform a function. Being able to recycle components to compute functions would also address another aspect to scaling self-assembling system, and would be another step towards creating systems using large numbers of components in the future.

8.4 Applications

In general, the advantages of three-dimensional self-assembly include the creation of structures that make a more efficient use of volume, as well as shorter interconnections between components (Whitesides & Grzybowski 2002). Specifically, the three-level approach is envisioned to being applicable to the design of structures (including at the nano and microscale), circuit fabrication, modular and swarm robotics control systems, synthetic biology, and DNA computing using self-assembly. For example, three-dimensional DNA computing has been proposed, but the required components have not been achieved at the time of writing (Pelletier & Weimerskrich 2002). The combination of component shape and an assembly protocol can be used to reduce component-to-component interaction errors and create more complex self-assembled structures using DNA nanotechnology (Turberfield 2011; Woo & Rothemund 2011). Furthermore, potential applications for three-dimensional self-assembly includes the creation of hybrid rapid prototyping technologies that make use of self-assembly at larger physical scales (Hiller & Lipson 2009), and potentially at the nanoscale (Gates et al. 2005).

8.5 Summary

A summary of the experimental results presented in this thesis was given first in this chapter. In total, six 2D experiments and eleven 3D experiments were conducted. The three-level approach for designing self-assembling systems via physically encoded information was used in the context of three design paradigms across the experiments: programming, evolving, and staging. A unique closed target structure with symmetric/asymmetric features was assigned to each experiment. With rule sets provided at level one, virtual testing was conducted at level two, and physical testing was conducted at level three. All the experiments successfully created their target structure at level two, with a statistical significance level of a p-value of 0. The successful level two results support the unique modelling features of the 2DcTAM/3DcTAM and the 2DscTAM/3DscTAM, namely: modelling self-assembly concurrently, not requiring seed tiles (components), permitting tile rotations, accommodating 2D and 3D components, and using an environment temperature of one. With the success of the level two experiments, a level three translation was conducted for all applicable systems. Each system was able to achieve its target structure. The physical experiments were statistically significant at either the 0.01 or the 0.05 level (meaning there is a 99% or a 95% chance the results were not due to chance), with the exception of one of the 3D staged experiments (3DSE3). Although one trial was successful in 3DSE3, the result is not considered statistically significant, where as all the other physical experiment are. Therefore, the successful 2D and 3D proof-of-concept experiments provide evidence to support the thesis hypothesis that it is possible to encode information as components and their corresponding environment to enable the self-assembly of closed target structures with desired morphologies.

Next, an example of how staging can be used to leverage a limited rule set, inspired by the use of a body plan in biological development, was provided. As future work, alternative physical information encoding schemes were discussed, including an example of a 9-magnetic-bit encoding scheme. This was followed with more broad directions of future work. These directions include all aspects to improve the three-level approach and the design paradigms used in the experiments (programming, evolving, and staging) to go beyond the proof-of-concept systems presented in this thesis, as well as the computational aspects of self-assembly, error correction methods, and scaling self-assembling systems. Finally, applications were discussed over a wide variety of disciplines and across physical scales to identify the potential practical benefits designing self-assembling systems via physically encoded information.

Chapter 9

Conclusions

Designing and constructing artificial self-assembling systems presents numerous challenges, and remains an elusive goal. In order to work towards achieving this goal, the aim of this thesis was to demonstrate the feasibility of a design methodology using physically encoded information – the three-level approach – to enable the self-assembly process resulting in closed target structures with symmetric/asymmetric features. Three design paradigms using the three level-approach were investigated in this thesis: programming, evolving, and staging. These three design paradigms further develop the notion of physical information, as information that is embodied as components in their corresponding environments that are conducive to the generation of physical processes, specifically in this thesis the process of self-assembly to construct target structures.

This chapter summarizes the preceding chapters, reviews the evidence satisfying the thesis hypothesis, provides the contributions of this work, lists the publications resulting from this research, and finally presents the conclusions of this thesis.

9.1 Thesis Summary

Chapter 1 provided a brief introduction to self-assembling systems, by providing two examples from nature where physical information enables the self-assembly of static structures (the crystalline growth of snow crystals) and dynamic machines (the development of the bacterial flagellum providing bacteria motility). It also provided scientific motivation to studying self-assembling systems, focusing on the potential benefits using self-assembly as an enabling technology for the creation of artificial systems. This chapter also described how the work presented in this thesis is expected to work towards creating more sophisticated self-assembling system by developing a self-assembly design process that allows for the encoding of physical information.

Chapter 2 provided a critical review of the relevant literature to this thesis. Physical self-assembling systems, from the macroscale, across the mesoscale, and down to the nanoscale presented the advantages and limitations to current physical, artificial system representing the state of the art. Limitations of current physical self-assembling systems include the prevention/reduction of errors during the self-assembly process, exclusion of varied component rotation interactions for the creation of 3D systems, and the limitation of methods to create target structures with symmetric/asymmetric features. DNA computing was used to underpin the theoretical aspects of self-assembly. The pioneering development of the aTAM (2D tile-based model capable of Turing universal computation at temperature two) was contrasted to its extensions, including recent developments in 2D and 3D tile-based self-assembly at a more easily physical implementable temperature of one. The aTAM was also used to describe how self-assembly has been proven to being an algorithmically NP-complete problem. Lastly, tile-based models and a tile-based physical system that used either Kolomogrov information or Shannon information were used to illustrate the limitation of both approaches to encapsulating physical self-assembling system, as both fundamentally disconnect information from matter and energy. The chapter concluded by contrasting top-down and bottom-up self-assembly design methodologies. In particular, evolutionary computing (a bottom-up algorithm) is well-suited to solving NP-complete problems.

Chapter 3 presented physical information analysis in the context of two self-assembling systems. The first system was an idealized model consisting of only unit spherical components in a fixed environment. Components could assemble into different open structures, such as cubic, hexagonal, and layered hexagonal lattices by varying the sticky site locations on components. Due to the homogenous component shapes, two types of component interaction rules (based on localized message passing forming gradients between components in a structure) using global axes information and axes count information were developed to create 2D and 3D closed target structures with symmetric/asymmetric features. Furthermore, these two types of component interaction rules were used to show how symmetric features in the target structures could be exploited to reduce the number of rules required. The second system was the first artificial, mechanical self-assembling system, an analogue to self-reproduction akin to molecular amplification in the form of templated self-assembly. An abstract description of the component assembly, component rotation, environment interactions, and autocatalysis rules based on the physical characteristics of the system was presented. Through reverse engineering, an equivalent physical system was constructed and successfully tested the proposed rules, and verified that the original system truly is a mechanical analogue to self-reproduction.

Chapter 4 introduced a self-assembly design methodology, based on a combination of the idealized model and the first mechanical analogue to self-assembly, referred to as the three-level approach. The three-level approach was inspired by the central dogma of molecular biology, the transfer of genetic information to create physical self-assembling shapes. The three levels - (1) definition of rule set, (2) virtual execution of rule set, and (3) physical realization of rule set - were described in the context of 2D and 3D self-assembly. The three-level approach provides a bottom-up method for designing physical self-assembling systems, by being able to directly map a set of rules to a physical system. At level one, an abstract, high-level set of rules were presented that describe the type of self-assembling systems used in this work. The 2DcTAM and the 3DcTAM, extensions to the aTAM by allowing for concurrent self-assembly and rotations, provided a Turing universal tile-based model using a temperature of one for the creation of 2D and 3D target structures respectively, at level two. Finally at level three, a component design space was used. The design space consisted of a key-lock-neutral shape space inspired by molecular recognition, and an assembly protocol space using either a 3-magnetic-bit pattern or a 5-magnetic-bit pattern for 2D and 3D components respectively. Furthermore, two 2D and two 3D physical encoding schemes were presented which leveraged different aspects of the design space to prevent/reduce errors during the self-assembly process and realize 3D component-to-component rotation interactions. Components were constructed using rapid prototyping, and placed in their corresponding, automated environments in 2D (on the surface of a tray) or in 3D (in a jar of fluid) providing vibrational energy to the mechanical components.

Chapter 5 was used to test the second 2D and second 3D physical component information encoding schemes, and their corresponding environments, in the context of a programming paradigm. A structure can be considered as the output of a physical program, connecting computation and self-assembly. Hand-designed rules were used to create three 2D and five 3D closed target structures with symmetric/asymmetric features, in a series of experiments. Each experiment was successful in the system being able to self-assemble its uniquely assigned target structure. Therefore, the three-level approach is a valid self-assembly design method, and the 2D and 3D physical information encoding schemes presented are capable of self-assembly.

Chapter 6 incorporated evolutionary computing into the three-level approach. Evolutionary computing was used to evolve rules (sets of components) to demonstrate a bottom-up method to designing physical self-assembling systems, which is computationally suitable for addressing self-assembly being an algorithmically NP-complete problem. Five experiments (two experiments in 2D and three experiments in 3D) were used to test the automated generation of component sets, and their ability to construct closed target structures with symmetric/asymmetric features in their corresponding environments. The successful experiments confirm the validity in using evolutionary computing to generate sets of components that can be mapped using physically encoded information, to create physical self-assembling systems.

Chapter 7 extended the three-level approach by incorporating staging, a method for dividing the self-assembly process into time intervals, inspired by staged biological development. Staged development in nature, e.g. the formation of a body plan, allows for the creation of more complex phenotypes, not otherwise possible. One important challenge when creating artificial self-assembling systems is caused by the use of components that lack the plasticity of biological cells. Using components that cannot differentiate results in self-assembly being constrained to a limited set of components and their bonding mechanisms. The benefit of staging is that it encodes the construction of a target structure in the staging algorithm itself, and not necessarily in the design of the components. One 2D and three 3D staged experiments were conducted. The successful results validate how physical features in a set of heterogeneous, passive, mechanical components can be exploited to reduce potential assembly errors, leverage rotational bonding mechanisms, and create closed target structures with symmetric/asymmetric features.

Chapter 8, lastly, summarized the results of the proof-of-concept experiments using the three design paradigms based on the three-level approach using physically encoded information, provided analysis of a more advanced staging technique, and offered future directions extending the research corresponding to the experiments. An example using an abstract dog target structure was used to show the formation of a body plan using staging, where component physical information could be reused at later time intervals, in the self-assembly of a single or multiple dog target structures. Higher-order 3D component physical information in the form of a 9-magnetic-bit pattern was presented to show its advantages to address rotation interaction errors during the self-assembly process. Broad directions of future work were explored, including improving all aspects of the three-level approach, error correction methods, and scaling self-assembling systems. As well, applications of the research described in this thesis were presented, including using a component design space consisting of component shape and an assembly protocol as physical information to enable the self-assembly of more complex DNA-based closed target structures with symmetric/asymmetric features.

9.2 Thesis Hypothesis Satisfaction

The hypothesis of this thesis was,

Thesis Hypothesis: it is possible to encode information as physical components and their corresponding environments to enable the self-assembly of closed structures with desired morphologies.

To encode information in this work means a process that transforms information into physical information. Here, the notion of physical information incorporates the concept of information embodied as components in a corresponding environment that are conducive to the generation of physical processes, specifically the process of self-assembly. As such, *to encode information* is a design problem, and requires a design solution.

Three self-assembly design questions were addressed to test the thesis hypothesis. These three self-assembly design problems required the creation of a self-assembly design methodology that allowed for an investigation into encoding physical information.

1. How to specify a set of rules that can be mapped to create a physical system using physically encoded information?

The three-level approach presented in *Chapter 4* provides a design methodology, where self-assembly rules can be translated to a physical system by mapping physically encoded information. The two component-to-component and component-to-environment interactions rules represent the smallest set of rules to describe the interactions required to create static self-assembling systems. These two rules are consistent with the definition of self-assembly, where the first rule incorporates component assembly, and the second rule incorporates selective component interactions. The three-level approach served as the basis for using evolution and staging, to address question two and question three.

2. How to automatically generate sets of rules using computer software, to address self-assembly being an algorithmically NP-complete problem?

Since the three-level approach can directly map a set of level one rules to a level three physical system – bottom-up design – evolutionary computing can be appropriately incorporated. Designing self-assembling systems is an NP-complete problem, and evolutionary computing is well-suited for addressing such problems. In *Chapter 6*, a proof-of-concept demonstration showed how the three-level approach can be used to evolve component sets. The benefit of using evolution to automatically generate sets of rules, is that only the functionality of a target structure is required, not its morphology.

3. How to use physical information to create more complex target structures?

Staging was used to create more complex target structures, not otherwise possible. In *Chapter 7*, a proof-of-concept demonstration was given where component sets were divided into two time intervals. The dog structure in *Chapter 8* served as an example for a more sophisticated staging technique, using the notion of a body plan.

Creating a self-assembly design methodology, the three-level approach, served as the basis for investigating these three self-assembly design problems. Each one demonstrated the ability of using the three-level approach to encode physical information, enabling the self-assembly process. Extensive virtual and physical experiments, both in 2D and in 3D, for each of these three questions were conducted. The successful experimental results provide strong statistical evidence to answer those three questions (Table 8.1 page 169, and Table 8.2 page 171). The experimental evidence supports the thesis hypothesis, as it is feasible to encode information as physical components and their environment to enable the self-assembly of closed structures with desired morphologies.

9.3 Thesis Contributions

The notion of using physical information to enable the self-assembly process is the principle contribution of this thesis. Detailed contributions resulting from the research presented in this thesis are the following:

- a framework for designing and creating self-assembling systems,
- examples of how component pattern formation can be exploited in homogenous systems using simple forms of gradient-based local communication between components to create 2D and 3D closed target structures with symmetric/asymmetric features,
- by reverse engineering, analysis and construction of the original L.S. Penrose and R. Penrose self-reproducing analogue,
- a bottom-up self-assembly design methodology, the three-level approach,

- minimal set of abstract, high-level self-assembly interaction rules for the creation of static self-assembling systems,
- temperature one tile-based models that incorporate component rotations to exploit symmetry, which are Turing universal (the 2DcTAM and the 3DcTAM), and which are extendible to include staging (the 2DscTAM and the 3DscTAM),
- a physical 2D component information encoding scheme that can reduce/prevent component interaction errors,
- a physical 3D component information encoding scheme that can reduce/prevent component interaction errors and encode component rotational properties,
- a programming design paradigm using physically encoded information,
- an evolutionary design paradigm to address self-assembly being an algorithmically NP-complete problem,
- a staging design paradigm using component physical information to reduce/prevent errors during the self-assembly process,
- new machining and raid prototyping techniques to fabricate components and construct their corresponding environments, and
- continued development to the notion of what is physical information.

9.4 Thesis Publications

The following is a list of publications resulting from the research conducted as a part of this thesis.

N. Bhalla and P.J. Bentley (in print). *Programming self-assembling systems via physically encoded information.* Invited Chapter in R. Doursat, H. Sayama, and O. Michel (Eds.) Morphogenetic Engineering. Springer.

N. Bhalla, P.J. Bentley, P.D. Vize, and C. Jacob (2011). *Programming and evolving self-assembling systems in three dimensions.* Natural Computing, Special Issue on Engineering Emergence. S. Stepney and P. Andrews (Eds.). DOI 10.1007/s11047-011-9293-6.

N. Bhalla, P.J. Bentley, P.D. Vize, and C. Jacob (2011). *Staging the self-assembly process using morphological information.* Proceedings of the European Conference on Artificial Life (ECAL 2011). p. 93-100.

N. Bhalla, P.J. Bentley, and C. Jacob (2010). *Evolving physical self-assembling systems in two-dimensions.* Proceedings of the International Conference on Evolvable Systems (ICES 2010). p.381-391.

N. Bhalla, P.J. Bentley, and C. Jacob (2007). *Mapping virtual self-assembly rules to physical systems.* Proceedings of the International Conference on Unconventional Computing (UC 2007). p. 117-148.

N. Bhalla and C. Jacob (2006). *A framework for analyzing and creating self-assembling systems.* Proceedings of the IEEE Symposium Series on Computational Intelligence, Swarm Intelligence Symposium (SIS 2006). p. 281-288.

9.5 Thesis Conclusions

This thesis investigated how physically encoded information can be used to design self-assembling systems. The three-level approach, where a set of rules can be mapped bottom-up to a physical system using physically encoded information, was used to demonstrate how it can be used within three paradigms: programming, evolving, and staging self-assembling systems. What this thesis has illustrated through these design paradigms is physical information can be used for the design of 2D and 3D closed target structures with symmetric/asymmetric features. Theoretical advancements were also made in this thesis, by developing tile-based models that are Turing universal at temperature one, incorporate component rotation in 2D and 3D, and model self-assembly concurrently. The theoretical advancements along with the 2D and 3D physical information encoding schemes used together in a self-assembly design methodology that can map a set of rules demonstrates the generation of physical processes aspect (specifically self-assembly) in advancing the notion of physical information. The successful experimental results, including strong statistical evidence, further supports the connection between self-assembly and computation. Designing self-assembling systems remains an elusive goal. However, the design advancements made in this thesis work towards solving this problem. The successful research presented in this thesis demonstrates a viable method for continued development in the area of self-assembly based on designing self-assembling systems via physically encoded information.

Bibliography

Adleman, L. (1994). Molecular computation of solutions to combinatorial problems. *Science*, *266*(5187), 1021–1024.

Adleman, L., Cheng, Q., Goel, A., & Huang, M. (2001). Running time and program size for self-assembled squares. In *ACM Symposium on Theory of Computing*, (pp. 740–758).

Adleman, L., Cheng, Q., Goel, A., Huang, M.-D., Kempe, D., de Espanés, P. M., & Rothemund, P. W. K. (2002). Combinatorial optimization problems in self-assembly. In *34th Annual ACM International Symposium on the Theory of Computing.*

Aggarwal, G., Cheng, Q., Goldwasser, M. H., Kao, M.-Y., de Espanés, P. M., & Schweller, R. T. (2005). Complexities for generalized models of self-assembly. *SIAM Journal on Scientific Computing*, *24*(6), 1493–1515.

Ampatzis, C., Tuci, E., Trianni, V., Christensen, A. L., & Dorigo, M. (2009). Evolving self-assembly in autonomous homogeneous robots: experiments with two physical robots. *Artificial Life*, *15*(4), 465–484.

Ball, P. (1994). The shape of things to come. *Nature*, *371*, 202–203.

Ball, P. (1999). *The self-made tapestry: pattern formation in nature.* Oxford, UK: Oxford University Press.

Ball, P. (2009a). *Branches, nature's patterns: a tapestry in three parts.* Oxford, UK: Oxford University Press.

Ball, P. (2009b). Feynman's fancy. *Chemistry World*, *6*(1), 58–62.

Ball, P. (2009c). *Flow, nature's patterns: a tapestry in three parts*. Oxford, UK: Oxford University Press.

Ball, P. (2009d). *Shapes, nature's patterns: a tapestry in three parts*. Oxford, UK: Oxford University Press.

Barish, R. D., Rothemund, P. W. K., & Winfree, E. (2005). Two computational primitives for algorithmic self-assembling: copying and counting. *Nano Letters*, *5*(12), 2586 – 2592.

Baum, R. (2003). Nanotechnology: drexler and smalley make the case for and against 'molecular assemblers'. *Chemical and Engineering News*, *81*(48), 37–42.

Beer, F. P., Johnston, E. R., Eisenberg, E., & Mazurek, D. (2009). *Vector mechanics for engineers: statics*. New York, NY: McGraw-Hill.

Bentley, P. J., & Wakefield, J. P. (1997). *Soft computing in engineering design and manufacturing*, chap. Generic evolutionary design. Springer Verlag.

Berger, R. (1966). The undecidability of the domino problem. *Memoirs of the American Mathematical Soceity*, *66*, 1 – 72.

Bernardini, F., & Gheorgh, M. (2004). Population p systems. *Journal of Universal Computer Science*, *10*(5), 509–539.

Bhalla, N. (2004). *Self-assembling systems in two-dimensions*. Master's thesis, University College London.

Bhalla, N. (2009). Book review: natalio krasnagor, steve gustafson, david a. pelta, and jose l. verdegay (eds.): systems self-assembly: multidisciplinary snapshots. *International Journal of Genetic Programming and Evolvable Machines*, *10*(4), 473–475.

Bhalla, N., & Bentley, P. J. (2006). Working towards self-assembling robots at all scales. In *3rd International Conference on Autonomous Robots and Agents*, (pp. 617–622).

Bhalla, N., & Bentley, P. J. (in print). *Morphogenetic engineering*, chap. Programming self-assembling systems via physically encoded information. Springer.

Bhalla, N., Bentley, P. J., & Jacob, C. (2007). Mapping virtual self-assembly rules to physical systems. In *International Conference on Unconventional Computing*, (pp. 117 – 148).

Bhalla, N., Bentley, P. J., & Jacob, C. (2010). Evolving physical self-assembling systems in two-dimensions. In *International Conference on Evolvable Systems*, (pp. 381 – 391).

Bhalla, N., Bentley, P. J., Vize, P. D., & Jacob, C. (2011a). Programming and evolving physical self-assembling systems in three dimensions. *Natural Computing*.

Bhalla, N., Bentley, P. J., Vize, P. D., & Jacob, C. (2011b). Staging the self-assembly process using morphological information. In *European Conference Artificial Life*, (pp. 93–100).

Bhalla, N., & Jacob, C. (2007). A framework for analyzing and creating self-assembling systems. In *IEEE Symposium Series on Computational Intelligence, Swarm Intelligence Symposium*, (pp. 281 – 288).

Binnig, G., Quate, C. F., & Gerber, C. (1986). Atomic force microscope. *Physical Review Letters*, *56*(9), 930–933.

Binnig, G., Rohrer, H., Gerber, C., & Weibel, E. (1982a). Surface studies by scanning tunneling microscopy. *Physical Review Letters*, *49*(1), 57–61.

Binnig, G., Rohrer, H., Gerber, C., & Weibel, E. (1982b). Tunneling through a controllable vacuum gap. *Applied Physics Letters*, *40*(2), 178–180.

Bishop, J., Burden, S., Klavins, E., Kreisberg, R., Malone, W., , & Nguyen, T. (2005). Programmable parts: a demonstration of the grammatical approach to self-organization. In *IEEE/RSJ International Conference on Intelligent Robots Systems*, (pp. 2644 - 2651).

Blasquez, I., & Poiraudeau, J.-F. (2003). Efficient processing of minkowski functionals on a 3d binary image using binary decision diagrams. *WSCG*, *11*(1).

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelliegnce: from natural to artificial systems*. Oxford University Press.

Boncheva, M., Bruzewicz, D. A., & Whitesides, G. M. (2003a). Formation of chiral, three-dimensional aggregates by self-assembly of helical components. *Langmuir*, *19*, 6066–6071.

Boncheva, M., Bruzewicz, D. A., & Whitesides, G. M. (2003b). Millimeter-scale self-assembly and its applications. *Pure and Applied Chemistry*, *75*(5), 621–630.

Boo, J. (2004). Sticky graphs - a tool to model self assembly. Tech. rep., Mid Sweden University, Faculty of Science, Technology and Media, Department of Engineering, Physics and Mathematics.

Bowden, N., Terfort, A., Carbeck, J., & Whitesides, G. M. (1997). Self-assembly of mesoscale objects into ordered two-dimensional arrays. *Science*, *276*(5310), 233–235.

Breivik, J. (2001). Self-organization of template-replicating polymers and the spontaneous rise of genetic information. *Entropy*, *3*(4), 273–279.

Brun, Y. (2008a). Constant-size tileset for solving an np-complete problem in nondeterministic linear time. In *DNA*, (pp. 26–35).

Brun, Y. (2008b). Solving np-complete problems in the tile assembly model. *Theoretical Computer Science*, *395*, 31–46.

Buchanan, A., Gazzola, G., & Bedau, M. (2008). Evolutionary design of a model of self-assembling chemical structures. In N. Krasnogor, S. Gustafson, D. A. Pelta, & J. L. Verdegay (Eds.) *Systems self-assembly: multidisciplinary snapshots*, vol. 5, chap. 4, (pp. 79–100). Amsterdam, NL: Elsevier.

Castano, A., Shen, W.-M., & Will, P. (2000). Conro: towards deployable robots with inter-robot metamorphic capabilities. *Autonomous Robots*, *8*(3), 309–324.

Chandran, H., Gopalkrishnan, N., & Reif, J. (2009). *Automata, languages and programming*, vol. 5555 of *Lecture Notes in Computer Science*, chap. The tile complexity of linear assemblies, (pp. 235–253). Berlin, DE: Springer-Verlag.

Chen, J., & Seeman, N. C. (1991). Synthesis from dna of a molecule with connectivity of a cube. *Nature*, *350*, 631–633.

Choi, S. I., Weck, M., Xu, B., Jeon, N. L., & Whitesides, G. M. (2000). Mesoscopic, templated self-assembly at the fluid-fluid interface. *Langmuir*, *16*, 2997–2999.

Christner, B. C., Morris, C. E., Foreman, C. M., Cai, R., & Sands, D. C. (2008). Ubiquity of biological ice nucleators in snowfall. *Science*, *319*(5867), 1214.

Clark, T. D., Ferrigno, R., Tien, J., Paul, K. E., & Whitesides, G. M. (2002). Template-directed self-assembly of 10-micron-sized hexagonal plates. *Journal of the American Chemical Society*, *124*(19), 5419–5426.

Clark, T. D., Tien, J., Duffy, D. C., Paul, K., & Whitesides, G. M. (2001). Self-assembly of 10-micron-sized objects into ordered three-dimensional arrays. *Journal of the American Chemical Society*, *123*(31), 7677–7682.

Committee to Review the National Nanotechnology Initiative, N. N. I., & National Research Council, N. R. C. (2006). *A matter of size: triennial review of the national nanotechnology iniative*, chap. Molecular self-assembly, (pp. 99–110). Washington, D.C.: The National Academic Press.

Cook, M., Fu, Y., & Schweller, R. (2011). Temperature 1 self-assembly: deterministic assembly in 3d and probablistic in 2d. In *ACM-SIAM Symposium on Discrete Algorithms*, (pp. 570–589).

Cook, S. (1971). The complexity of theorem-proving procedures. In *ACM Symposium on Theory of Computing*, (pp. 151–158).

Corne, D., & Knowles, J. (2007). Techniques for highly multiobjective optimization: some nominated points are better than others. In *Conference on Genetic and Evolutionary Computation*, (pp. 773–780).

Cox, D., & Snell, E. (1989). *Analysis of binary data.* Boca Raton, FL: Chapman and Hall/CRC Press.

Crick, F. H. C. (1970). Central dogma of molecular biology. *Nature*, *227*, 561–563.

Culik, K. (1996). An aperiodic set of 13 wang tiles. *Discrete Mathematics*, *160*, 245–251.

Culik, K., & Kari, J. (1996). An aperiodic set of wang cubes. In *Symposium on Theoretical Aspects of Computer Science*, vol. 1046, (pp. 137–146).

Damoto, R., Kawakami, A., & Hirose, S. (2001). Study of super-mechano colony: concept and basic experimental set-up. *Advanced Robotics*, *15*(4), 391–408.

Dawkins, R. (2004). *The ancestor's tale: a pilgrimage to the dawn of evolution.* New York, NY: Houghton Mifflin Company.

de Espanés, P. M., & Goel, A. (2007). Toward minimum self-assembled counters. In *International Conference on DNA Computing*, (pp. 46–53).

Demaine, E. D., Demaine, M. L., Fekete, S. P., Ishaque, M., Rafalin, E., Schweller, R. T., & Souvaine, D. L. (2008). Staged self-assembly: nanomanufacture of arbitrary shapes with o(1) glues. *Natural Computing*, *7*(3), 347–370.

Diller, E., Pawashe, C., Floyd, S., & Sitti, M. (2011). Assembly and disassembly of magnetic mobile micro-robots towards deterministic 2-d reconfigurable micro-systems. *International Journal of Robotics Research.*

Dittrich, P., Ziegler, J., & Banzhaf, W. (2001). Artificial chemistries - a review. *Artificial Life*, *7*, 225–275.

Dorigo, M., Tuci, E., Trianni, V., Groß, R., Nouyan, S., Ampatzis, C., Labella, T., O'Grady, R., Bonani, M., & Mondada, F. (2006). Swarm-bot: Design and implementation of colonies of self-assembling robots. In *Computational Intelligence: Principles and Practice*, (pp. 103–135). New York, NY: IEEE Computational Society.

Doty, D. (2009). Randomized self-assembly for exact shapes. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, (pp. 85–94).

Douglas, S. M., Deitz, H., Liedl, T., Högberg, B., Graf, F., & Shih, W. M. (2009). Self-assembly of dna into nanoscale three-dimensional shapes. *Nature*, *459*, 414–418.

Doursat, R. (2008). Programmable architectures that are complex and self-organized: from morphogenesis to engineering. In *Artificial Life XI*, (pp. 181–188).

Drexler, K. E. (1981). Molecular engineering: an approach to the development of general capabilities for molecular manipulation. *Proceedings of the National Academy of Sciences*, *78*(9), 5275–5278.

Drexler, K. E. (1986). *Engines of creation, the coming era of nanotechnology*. New York, NY: Anchor Press/Doubleday.

Drexler, K. E. (1992). *Nanosystems: molecular machinery, manufacturing, and computation*. New York, NY: Wiley, 1st ed.

Fernandez, J., & Khademhosseini, A. (2010). Micro-masonry: construction of 3d structures by microscale self-assembly. *Advanced Materials*, *22*, 1–4.

Feynman, R. P. (1960). There's plenty of room at the bottom: an invitation to enter a new field of physics. *Caltech Engineering and Science*, *23*(5), 22–36.

Feynman, R. P. (1993). Infinitesimal machinery. *Journal of Microelectricalmechanical Systems*, *2*(1), 4–14.

Frankel, F. C., & Whitesides, G. M. (2009). *No small matter: science on the nanoscale*. Cambridge, MA: The Belknap Press of Harvard University Press.

Friedman, B. R. (1998). *Physics from Fisher information: a unification*. Cambridge, UK: Cambridge University Press.

Friedman, B. R. (2001). Physics from fisher information. *Mathematics Today*, *37*, 115–119.

Frietas, R. A., & Merkel, R. C. (2004). *Kinematic self-replicating machines*. Georgetown, TX: Landes Bioscience.

Fukuda, T., Buss, M., Hosokai, H., & Kawauchi, Y. (1991). Cell structured robotic system cebot: control, planning and communication methods. *Robotics and Autonomous Systems*, *7*(2-3), 239–248.

Garcias, D. H., Kavthekar, V., Love, J. C., Paul, K. E., & Whitesides, G. M. (2002). Fabrication of millimeter-scale, patterned polyhedra by self-assembly. *Advanced Materials*, *14*(3), 235–238.

Garcias, D. H., Tien, J., Breen, T. L., Hsu, C., & Whitesides, G. M. (2000). Forming electrical networks in three dimensions by self-assembly. *Science*, *289*(5482), 1170–1172.

Garey, M., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of np-completeness*. San Francisco, CA: W.H. Freeman and Company.

Gates, B. D., Xu, Q., Stewart, M., Ryan, D., Wilson, C. G., & Whitesides, G. M. (2005). New approaches to nanofabrication: molding, printing, and other techniques. *Chemical Reviews*, *105*, 1171–1196.

Goel, A., Cheng, Q., & de Espanés, P. M. (2004). Optimal self-assembly of counters at temperature two. In *Foundations of Nanoscience: Self-Assembled Architectures and Devices*.

Goldstein, S. C., Campbell, J. D., & Mowry, T. C. (2005). Programmable matter. *IEEE Computer*, *38*(6), 99–101.

Griffith, S. T., Goldwater, D., & Jacobson, J. M. (2005). Self-replication from random parts. *Nature*, *437*(7059), 636.

Groot, R., & Warren, P. (1997). Dissipative particle dynamics: bridging the gap between atomistic and mesoscopic simulations. *Journal of Chemical Physics*, *107*, 4423–4435.

Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2006). Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, *22*(6), 1115–1130.

Groß, R., & Dorigo, M. (2008). Self-assembly at the macroscopic scale. *Proceedings of the IEEE*, *96*(9), 1490–1508.

Grzybowski, B., & Whitesides, G. (2002a). Dynamics aggregation of chiral spinners. *Science*, *296*(5568), 718–721.

Grzybowski, B., & Whitesides, G. (2002b). Three-dimensional dynamic self-assembly of spinning magnetic disks: vortex crystals. *The Journal of Physical Chemistry B*, *106*(6), 1188–1194.

Grzybowski, B. A., Stone, H. A., & Whitesides, G. M. (2000). Dynamic self-assembly of magnetized, millimeter-sized objects rotating at the liquid-air interface. *Nature*, *405*, 1033–1036.

Hawkes, E., An, B., Benberbou, N. M., Tanaka, H., Kim, S., Demaine, E. D., Rus, D., & Wood, R. J. (2010). Programmable matter by folding. *Proceedings of the National Academy of Sciences*, *107*(28), 12441–12445.

He, Y., Ye, T., Su, M., Zhang, C., Ribbe, A. E., Jiang, W., & Mao, C. (2008). Heirarchical self-assembly of dna into symmetric supramolecular polyhedra. *Nature*, *452*, 198–201.

Hiller, J., & Lipson, H. (2009). Design and analysis of digital materials for physical 3d voxel prinitng. *Rapid Prototyping Journal*, *15*(2), 137–149.

Hirose, S. (2001). Super mechano-system: new perspective for versatile robotic system. In *International Symposium on Experimental Robotics*, vol. 271, (pp. 249–258).

Hirose, S., Damoto, R., & Kawakami, A. (2000). Study of super-mechano-colony (concept and basic experimental setup). In *IEEE/RSJ International Conference on Intelligent Robots Systems Conf. Intell. Robots Sys.*, vol. 3, (pp. 1664–1669).

Hosokawa, K., Shimoyama, I., & Miura, H. (1996). Dynamics of self-assembling systems: analogy with chemical kinetics. *Artificial Life*, *57*(2), 117–125.

Ingber, D. E. (1998). The architecture of life. *Scientific American*, *278*(1), 48–57.

Ishiguro, K. (2004). Revealing the mystery of the bacterial flagellum: a self-assembling nanomachine with fine switching capability. *Japan Nanonet Bulletin.*

Jacob, C. (2001). *Illustrating evolutionary computing with mathematica.* San Francisco, CA: Morgan Kauffman Publishers.

Jacobson, H. (1958). On models of reproduction. *American Scientist*, *46*, 255–284.

Jones, C., & Matari, M. J. (2003). From local to global behavior in intelligent self-assembly. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, vol. 1, (pp. 721–726).

Jones, R. A. L. (2004). *Soft machines: nanotechnology and life.* Oxford, UK: Oxford University Press.

Jonoska, N., & McColm, G. L. (2009). Complexity classes for self-assembling flexible tiles. *Theoretical Computer Science*, *410*, 332–346.

Kaewkamnerdpong, B. (2008). *Modelling nanorobot control using swarm intelligence.* Ph.D. thesis, University College London.

Kaewkamnerdpong, B., Bhalla, N., & Bentley, P. J. (2007). *Advances in Computers: Nanotechnology*, vol. 71, chap. Programming nanotechnlogy: learning from nature, (pp. 2–34). Amsterdam, NL: Academic Press.

Kalontarov, M., Tolley, M. T., Lipson, H., & Erickson, D. (2010). Hydrodynamically driven docking of blocks for 3d fluidic assembly. *Microfluids and Nanofluids*, *9*, 551–558.

Kao, M. Y., & Schweller, R. T. (2008). Randomized self-assembly for approximate shapes. In *Proceedings of the 35th Intiernational Colloquium on Automata, Languages and Programming*, (pp. 370–384).

Kari, L., & Mahalingam, K. (2010). Watson-crick palindromes in dna computing. *Natural Computing*, *9*, 297–316.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Proceedings of the Symposium on the Complexity of Computer Computations*, (pp. 85–103).

Kauffman, S. (2000). *Investigations*. Oxford, UK: Oxford University Press.

Kinsella, J. M., & Ivanisevic, A. (2005). Enzymatic clipping of dna wires coated with magnetic nanoparticles. *Journal of the American Chemical Society*, *127*(10), 3276–3277.

Klavins, E., Ghrist, R., & Lipsky, D. (2006). A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automated Control*, *51*, 949–962.

Klein, J. (2002). Breve: a 3d simulation environment for the simulation of decentralized systems and artificial life. In *Artificial Life VIII, 8th International Conference on the Simulation and Synthesis of Living Systems*, (pp. 329–334).

Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *International Journal of Computer Mathematics*, *1*(1), 1–7.

Krasnogor, N., Gustafson, S., Pelta, D. A., & Verdegay, J. L. (Eds.) (2008). *Systems self-assembly: multidisciplinary snapshots*, vol. 5. Amsterdam, NL: Elsevier.

Krishnan, M., Tolley, M. T., Erickson, D., & Lipson, H. (2009). Hydrodynamically tunable fluidic affinities for fluidic assembly. *Langmuir*, *25*, 3769–3774.

Krishnan, M., Tolley, M. T., Lipson, H., & Erickson, D. (2008). Increased robustness for fluidic self assembly. *Physics of Fluids*, *20*(073304), 1–16.

Landauer, R. (1996). The physical nature of information. *Physics Letters A*, *217*, 188–193.

Landauer, R. (1999). Information is a physical entity. *Physica A*, *263*(1), 63–67.

Li, L., Krasnogor, N., & Garibaldi, J. (2006). Automated self-assembly programming paradigm: initial investigation. In *IEEE International Workshop on Engineering of Autonomic and Autonomous Systems*, (pp. 25–34).

Li, L., Siepmann, P., Smaldon, J., Terrazas, G., & Krasnogor, N. (2008). Automated self-assembling programming. In N. Krasnogor, S. Gustafson, D. A. Pelta, & J. L. Verdegay (Eds.) *Systems self-assembly: multidisciplinary snapshots*, vol. 5, chap. 13, (pp. 281–307). Amsterdam, NL: Elsevier.

Libbercht, K. G. (2001). Morphogenesis on ice: the physics of snow crystals. *Engineering and Science*, *LXIV*(1), 10–19.

Macnab, R. M. (2003). How bacteria assemble flagella. *Annual Review of Microbiology*, *57*, 77–100.

Mandelbrot, B. B. (1977). *The fractal geometry of nature*. Trenton, NJ: W. H. Freeman.

Mao, C., Thalladi, V. R., Wolfe, D. B., Whitesides, S., & Whitesides, G. M. (2002). Dissections: self-assembled aggegates that spontaneously reconfigure their structure when their environment changes. *Journal of the American Chemical Society*, *124*(49), 14508–14509.

Mathews, N. (2008). *Self-replication and swarm robotics: a seed-based approach using autonomous robots*. Master's thesis, University of Freiburg.

Mathews, N., Christensen, A. L., O'Grady, R., & Dorigo, M. (2010). Cooperation in a heterogeneous robot swarm through spatialy targeted communication. In *International Conference on Ant Colony Optimization and Swarm Intelligence*, (pp. 400–407).

Meng, Y., & Kashyap, N. (2009). Controlling errors in the process of molecular self-assembly. In *Communications, Control, and Computing*, (pp. 676–677).

Michielsen, K., & de Raedt, H. (2000). Morphological image analysis. *Computer Physics Communications*, *132*, 94–103.

Mitchell, M. (1996). *An introduction to genetic algorithms*. Cambridge, MA: The MIT Press.

Mitchell, M. (2006). Complex systems: network thinking. *Artificial Intelligence*, *170*(18), 1194–1212.

Mitchell, M. (2009). *Complexity: a guided tour*. Oxford, UK: Oxford University Press.

Miyashita, S., Casanova, F., Lungarella, M., & Pfeifer, R. (2010). *Cutting Edge of Robotics*, chap. Peltier-Based Freeze-Thaw Connector for Waterborne Self-Assembly Systems, (pp. 187–198). Rijeka, HR: InTech.

Miyashita, S., Nagy, Z., Nelson, J. B., & Pheifer, R. (2009). The influence of shape on parallel self-assembly. *Entropy*, *11*, 643–666.

Mondada, F., Guignard, A., Bonani, M., Bär, D., Lauria, M., & Floreano, D. (2003). Swarm-bot: from concept to implementation. In *IEEE/RSJ International Conference on Intelligent Robots Systemsernational Conference on Intelligent Robots Systems Conf. Intell. Robots Sys. Conf. Intell. Robots Syst.*, vol. 2, (pp. 1626–1631).

Moore, E. (1962). Machine models of self-replication. In *Symposium in Applied Mathematics*, (pp. 17–33).

Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., & Kokaji, S. (2002). M-tran: self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, *7*(4), 431–441.

Murphy, M. P., & O'Neill, L. A. J. (1997). *What is life? the next fifty years: speculations on the future of biology*. Cambridge, UK: Cambridge University Press.

Nagpal, R. (2006). Self-organizing shape and pattern: from cells to robots. *IEEE Intelligent Systems*, *21*, 50–53.

Nalwa, H. S. (Ed.) (2004). *Encyclopedia of nanoscience and nanotechnology*. Valencia, CA: American Scientific Publishers, 1st ed.

Nouyan, S., Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2006). Group transport along a robot chain in a self-organized robot colony. In *International Conference on Intelligent Austonomous Systems*, (pp. 433–442).

O'Grady, R., Groß, R., Mondada, F., Bonanin, M., & Dorigo, M. (2005). Self-assembly on demand in a group of physical autonomous mobile robots navigating rough terrain. In *European Conference Artificial Life*, (pp. 272–281).

Olson, A., Hu, Y., & Keinan, E. (2007). Chemical mimicry of viral capsid self-assembly. *Proceedings of the National Academy of Sciences*, *104*(52), 20731–20736.

Omabegho, T., Sha, R., & Seeman, N. C. (2009). A bipedal dna brownian motor with coordinated legs. *Science*, *324*(5923), 67–71.

Patitz, M. J., Schweller, R. T., & Summers, S. M. (2011). Exact shapes and turing universailty at temperature 1 with a single negative glue. In *International Conference on DNA Computing and Molecular Programming*, (pp. 175–189).

Pauling, L. (1935). The structure and entropy of ice and other crystals with some randomness of atomic arrangement. *Journal of the American Chemical Society*, *57*(12), 2680–2684.

Păun, G., Rozenberg, G., & Salomaa, A. (1998). *DNA computing - new computing paradigms*. Berlin, DE: Springer.

Pelesko, J. (2007). *Self assembly: the science of things that put themselves together*. Boca Raton, FL: Chapman and Hall/CRC Press.

Pelletier, O., & Weimerskrich, A. (2002). Algorithmic self-assembly of dna tiles and its applications to cryptanalysis. In *Conference on Genetic and Evolutionary Computation*, (pp. 139–146).

Penrose, L. (1958). Mechanics of self-reproduction. *Annals of Human Genetics*, *23*, 59–72.

Penrose, L. (1959). Self-reproducing machines. *Scientific American*, *200*, 105–114.

Penrose, L. S., & Penrose, R. (1957). A self-reproducing analogue. *Nature*, *179*(4571), 1183–1184.

Penrose, R. (1974). Role of aesthetics in pure and applied research. *Bulletin of the Institute of Mathematics and its Applications*, *10*, 266.

Prusinkiewicz, P., & Linenmayer, L. (1990). *The algorithmic beauty of plants*. Berlin, DE: Springer.

Reiffel, J., & Pollack, J. (2005). Evolutionary fabricatiom: the emergence of novel assembly methods in artificial ontogenies. In *Conference on Genetic and Evolutionary Computation: Scalable, Evolvable, Emergent Design and Developmental Systems Workshop*, (pp. 265–272).

Rickwood, D., & Lund, V. (1998). Attachment of dna and oligonucleotides to magnetic particles: methods and applications. *Fresenius' Journal of Analytical Chemistry*, *330*(4-5), 330–330.

Rothemund, P. W. K. (2000). Using lateral capillary forces to compute by self-assembly. *Proceedings of the National Academy of Sciences*, *97*(3), 984–989.

Rothemund, P. W. K. (2005). Design of dna origami. In *International Conference on Computer-Aided Design*, (pp. 471–478).

Rothemund, P. W. K. (2006). Folding dna to create nanoscale shapes and patterns. *Nature*, *440*, 297–302.

Rothemund, P. W. K., Papadakis, N., & Winfree, E. (2004). Algorithmic self-assembly of dna sierpinski triangles. *Public Library of Science Biology*, *2*(12), 2041–2053.

Rothemund, P. W. K., & Winfree, E. (2000). The program size complexity of self-assembled squares. In *ACM Symposium on Theory of Computing*, (pp. 459–468).

Schrödinger, E. (1944, reprinted 2003). *What is life? with mind and matter and autobiographical sketches*. Canto Series. Cambridge, UK: Cambridge University Press.

Seeman, N. C. (2004). Nanotechnology and the double helix. *Scientific American*, *290*(6), 64–75.

Seeman, N. C. (2007). An overview of structural dna nanotechnology. *Molecular Biology*, *3*(37), 246–257.

Senechal, M. (2006). What is a quasicrystal? *Notice of the AMS*, *53*(8), 886–887.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technical Journal*, *27*, 379–423 and 623–656.

Shechtman, D., Belch, I., Gratias, D., & Cahn, J. (1984). Metallic phase with long-range orientational order and no translational symmetry. *Physical Review Letters*, *53*(20), 1951.

Sipser, M. (1997). *Introduction to the theory of computation.* Boston, MA: PWS Publishing Company.

Soille, P. (2003). *Morphological Image Analysis.* Berlin, DE: Springer.

Solveichik, D., & Winfree, E. (2007). Complexity of self-assembled shapes. *SIAM Journal on Computing*, *36*(6), 1544–1569.

Stepney, S., Braustein, S. L., Clark, J. A., Tyrrell, T., Adamatzky, A., Smith, R. E., Addis, T., Johnson, C., Timmis, J., Welch, P., Milner, R., & Partridge, D. (2005). Journeys in non-classical computation i: a grand challenge. *Parallel Emergent and Distributed Systems*, *20*(1), 5–19.

Stepney, S., Braustein, S. L., Clark, J. A., Tyrrell, T., Adamatzky, A., Smith, R. E., Addis, T., Johnson, C., Timmis, J., Welch, P., Milner, R., & Partridge, D. (2006). Journeys in non-classical computation ii: initial journeys and waypoints. *Parallel Emergent and Distributed Systems*, *21*(2), 97–125.

Stewart, I. (1995). Four encounters with sierpinski's gasket. *Mathematical Intelligencer*, *17*(1), 52–64.

Symonds, N. (1986). What is life? schrödinger's influence on biology. *The Quarterly Review of Biology*, *61*(2), 221–226.

Symonds, N. (1987). Schrödinger and what is life? *Nature*, *327*, 663–664.

Terfort, A., Bowden, N., & Whitesides, G. M. (1997). Three-dimensional self-assembly of millimeter-scale components. *Nature*, *386*, 162–164.

Terrazas, G., Gheorghe, M., Kendall, G., & Krasnogor, N. (2007). Evolving tiles for automated self-assembly design. In *IEEE Congress on Evolutionary Computation*, (pp. 2001–208).

Thompson, D. W. (1917). *On growth and form*. Cambridge, UK: Dover Publications.

Tibbits, S. (2010). *Logic matter: digital logic as heuristics for physical self-guided-assembly*. Master's thesis, Massachusetts Institute of Technology.

Tolley, M. T., Krishnan, M., Erickson, D., & Lipson, H. (2008). Dynamically programmable fluidic assembly. *Applied Physics Letters*, *93*(254105), 1–3.

Tolley, M. T., & Lipson, H. (2010). On-line assembly planning for stochastically reconfigurable systems. *International Journal of Robotics Research*, *30*(13), 1566–1584.

Toumey, C. (2005). Apostolic succession: does nanotechnology descend from richard feynman's 1959 talk? *Engineering and Science*, *LXVIII*(1-2), 16–23.

Turberfield, A. (2011). Dna nanotechnology: geometrical self-assembly. *Nature Chemistry*, *3*, 580–581.

Turberfield, A. J., Mitchell, J. C., Yurke, B., A. P. Mills, J., Blakey, M. I., & Simmel, F. C. (2003). Dna fuel for free-running nanomachines. *Physical Review Letters*, *90*(11), 118102:1–118102:4.

Turing, A. (1936). On computable numbers, with application to the entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2*, *42*, 230–265.

Vieira, F., & Barbosa, V. (2011). Optimization of supply diversity for the self-assembly of simple objects in two and three dimensions. *Natural Computing*, *10*(1), 551–581.

Wang, H. (1961). Proving theorems by pattern recognition. *Bell Systems Technical Journal*, *40*(1), 1–41.

Wang, H. (1965). Games, logic and computers. *Scientific American*, *213*(5), 98–106.

Watson, J. D., & Crick, F. H. C. (1953). Molecular structure of nucleic acids - a structure for deoxyribose nucleic acid. *Nature*, *171*(4356), 737–738.

White, P., Kopanski, K., & Lipson, H. (2004). Stochastic self-reconfigurable cellular robotics. In *IEEE International Conference on Robotics and Automation*, (pp. 2888–2893).

White, P., Zykov, V., Bongard, J., & Lipson, H. (2005). *Three dimensional stochastic reconfiguration of modular robots*, (pp. 161–168). Cambridge, MA: MIT Press.

Whitesides, G. M. (2001). The once and future nanomachine: biology outmatches futurists' most elaborate fantasies for molecular robots. *Scientific American*, *285*(3), 78–83.

Whitesides, G. M., & Boncheva, M. (2002). Beyond molecules: self-assembly of mesoscopic and macroscopic components. *Proceedings of the National Academy of Sciences*, *99*(8), 4769–4774.

Whitesides, G. M., & Boncheva, M. (2005). Making things by self-assembly. *Materials Research Society Bulletin*, *30*, 736–742.

Whitesides, G. M., & Grzybowski, B. (2002). Self-assembly at all scales. *Science*, *295*(5564), 2418–2421.

Whitesides, G. M., & Love, J. C. (2001). The art of building small. *Scientific American*, *285*(3), 38–47.

Winfree, E. (1995). On the computational power of dna annealing and ligation. *DNA Based Computers*, *27*, 199–221.

Winfree, E. (1998a). *Algorithmic self-assembly of dna.* Ph.D. thesis, California Institute of Technology.

Winfree, E. (1998b). Simulations of computing by self-assembly. In *4th International Meeting on DNA Based Computing.*

Winfree, E. (1999). Algorithmic self-assembly of dna: theoretical motivations and 2d assembly experiments. *Journal of Biomolecular Structure and Dynamics*, *11*(2), 263–270.

Winfree, E., Liu, F., Wenzier, L. A., & Seeman, N. C. (1998a). Design and self-assembly of two-dimensional dna crystals. *Nature*, *394*(6), 539–544.

Winfree, E., Yang, X., & Seeman, N. C. (1998b). Universal computation via self-assembly of dna: some thoery and experiments. *DNA Based Computers*, *II*, 191–213.

Witten, T. A., & Sander, L. M. (1981). Diffusion-limited aggregation - a kinetic criticial problem. *Physical Review Letters*, *47*, 1400–1403.

Wolfram, S. (2002). *A new kind of science.* Champaign, IL: Wolfram Media.

Wolpert, L. (1998). *The principles of development.* Oxford, UK: Oxford University Press.

Woo, S., & Rothemund, P. W. K. (2011). Programmable molecular recognition based on the geometry of dna nanostructures. *Nature Chemistry*, *3*(829), 620–727.

Wu, H., Thalladi, V. R., Whitesides, S., & Whitesides, G. M. (2002). Using hierarchical self-assembly to form three-dimensional lattices of spheres. *Journal of the American Chemical Society*, *124*(48), 14495–14502.

Yim, M., Duff, D. G., & Roufas, K. D. (2000). Polybot: a modular reconfigurable robot. In *IEEE International Conference on Robotics and Automation*, vol. 1, (pp. 514–520).

Yonekura, K., Maki, S., Morgan, D. G., DeRosier, D. J., Vonderviszt, F., Imada, K., & Namba, K. (2000). The bacterial flagellar cap as the rotary promoter of flagellin self-assembly. *Science*, *290*(5499), 2148–2152.

Yurke, B., Turberfield, A. J., Mills, A. P., Simmel, F. C., & Neumann, J. L. (2000). A dna-fuelled molecular machine made of dna. *Nature*, *406*(605).

Zhang, D. Y., & Seelig, G. (2011). Dynamic dna nanotechnology using strand-displacement reactions. *Nature Chemistry*, *3*, 103–113.

Zhang, X., Lin, M., & Niu, Y. (2011). Application of 3d dna self-assembly for graph coloring problem. *Journal of Computational and Theoretical Nanoscience*, *8*(10), 2042–2049.

Zhang, Y., & Seeman, N. C. (1994). Construction of dna-truncated octahedron. *Journal of the American Chemical Society*, *5*(116), 1661–1669.

Zykov, V., Mytillinaios, E., Adams, B., & Lipson, H. (2005). Robotics: self-reproducing machines. *Nature*, *435*, 163–164.

Zykov, V., Mytillinaios, E., Desnoyer, M., & Lipson, H. (2007). Evolved and designed self-reproducing modular robotics. *IEEE Transactions on Robotics*, *23*(2), 308–319.

Zykov, V., Williams, P., Lassabe, N., & Lipson, H. (2008). Molecubes extended: diversifying capabilities of open-source modular robotics. In *Intelligent Robots and Systems: Self-Reconfigurable Robots, Systems and Applications Workshop.*

Appendix A

Component Specifications

The following is a list of the materials and methods used for constructing the physical components corresponding to the original Penrose system (Penrose & Penrose 1957) presented in *Chapter 3: Physical Information Analysis*, and the 2D and 3D experiments presented in *Chapter 5: Programming Self-Assembling Systems*, *Chapter 6: Evolving Self-Assembly Rule Sets*, and *Chapter 7: Staging the Self-Assembly Process.*

A.1 Penrose Components

The materials used for constructing the Penrose components include:

- Brass plate (7/64"; approximately 2.78 mm)
- CNC Takumi V6 3-axis milling machine
- Rhino3D version 4.0 computer-aided design (CAD) software

The method for constructing the Penrose components has the following two steps:

1. Create the CNC files using Rhino3D for the α and β components, based on the specifications in Figure A.1.
2. Machine four α and four β components using a CNC Takumi V6 3-axis milling machine with the brass plate and CNC files.

A.2 2D Components

The materials used for constructing the 2D components include:

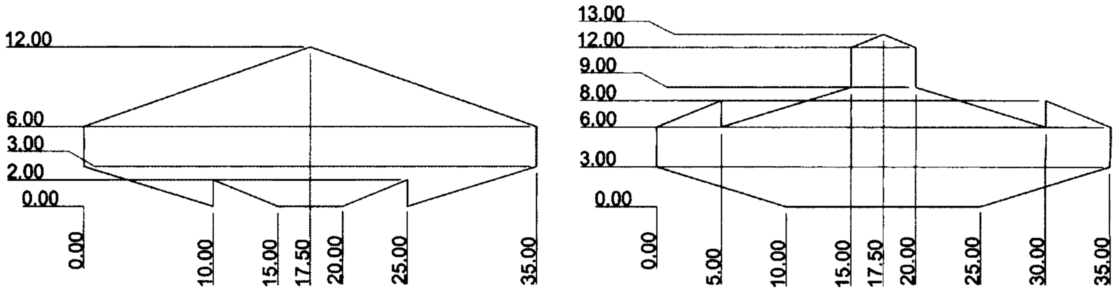


Figure A.1: Mathematical specifications for the Penrose components α (left) and β (right).

- Eden 333 Polyjet rapid prototyping machine
- Vero Gray resin
- Neodymium (NdFeB) disc magnets; 1/16" × 1/32" (diameter × height), grade N50
- Magnetic pole identifier
- Vice; built from a PanaVise bench clamp mount, a PanaVise low profile base, and a PanaVise low profile head
- Sharpie paint pens; oil based, extra fine point (colours: red and blue)
- Rhino3D version 4.0 CAD software

The method for constructing the 2D components has the following eight steps:

1. Create the CAD files using Rhino3D for the 2D magnet placement tool (Figure A.2, page 227), for the 2D component protection tool (Figure A.3, page 227) and for the 2D components, based on the specifications in Figure A.4 (page 228) and in association with the 2D component shape space (to determine key, lock, and neutral information locations on each 2D component, as well as any component interaction markers on the top surfaces of the 2D components).

2. Fabricate two 2D magnet placement tools and the 2D components using an Eden 333 Polyjet rapid prototyping machine with Vero Gray resin and the CAD files.

3. Insert three neodymium disc magnets in each 2D magnet placement tool, to create one tool with magnetic north polarity and the other with magnetic south polarity (identify polarity using the magnetic pole identifier).

4. Paint the 2D magnet placement tools, using the Sharpie paint pens (blue for magnetic north and red for magnetic south), to complete the construction of the 2D magnet placement tools.

5. Insert magnets into the 2D components by first identifying the appropriate 3-magnetic-bit pattern for each key and lock, and then placing magnets on the appropriate 2D magnet placement tool (two magnets for a key shape and three magnets for a lock shape) and using the vice to insert the magnets into the appropriate location in the 3-magnetic bit pattern (Figure A.5, page 228).

6. Remove the 2D magnet placement tool and 2D component from the vice and separate the 2D component placement tool and the 2D component (the extra magnet will dislodge and create an air gap on the 2D component where the magnets were inserted; this follows the 2D component physical encoding scheme for minimizing key-to-lock error interactions provided in Table 4.5 (page 97).

7. Repeat steps seven and eight until all magnets have been inserted into all the 2D components.

8. Paint, using the Sharpie paint pens, the 2D component interaction markers (refer to Table 4.5, page 97, to use the appropriate colours) corresponding to the 3-magnetic-bit patterns used, to complete the construction of the 2D components.

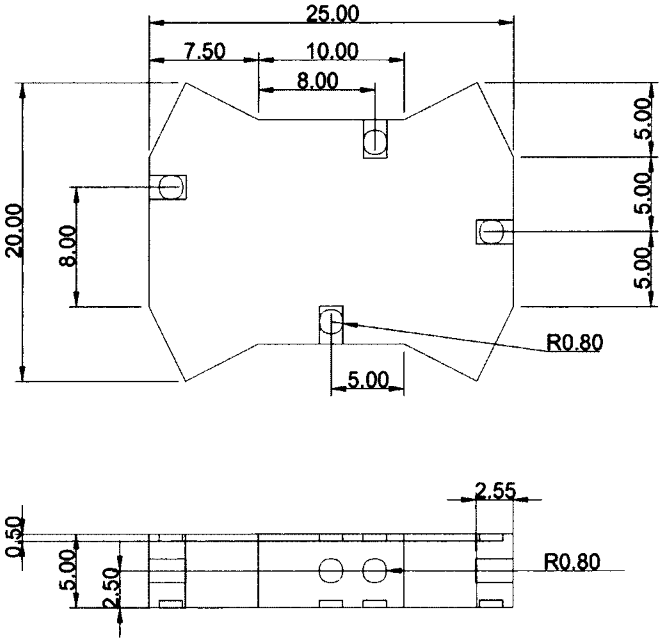


Figure A.2: 2D magnet tool specification (top/bottom and front view) - all magnet holes on the sides have the same radius (depth at key sites is 1.35 millimetres, and depth at lock sites is 2.20 millimetres), and all paint holes on the top an bottom have the same radius (all construction units in millimetres).

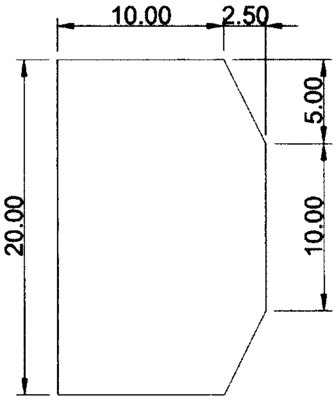


Figure A.3: 2D component protection tool (top view) - the height of this part is 5 millimetres (all construction units in millimetres).

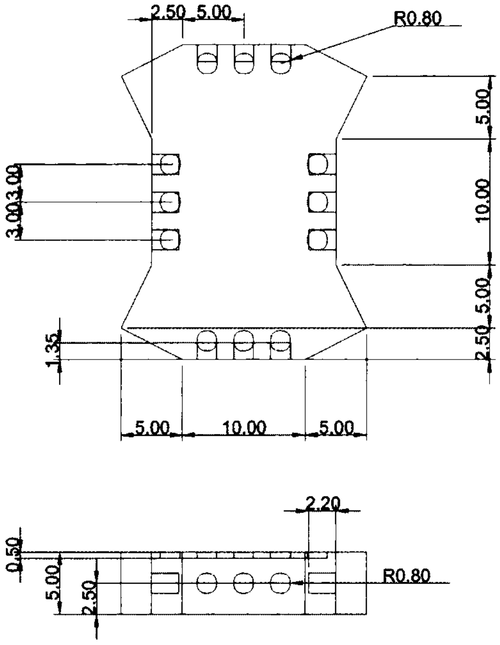


Figure A.4: 2D component specification (top view and front view) - all magnet holes on the sides have the same radius, and all paint holes on the top have the same radius and depth (all construction units in millimetres).

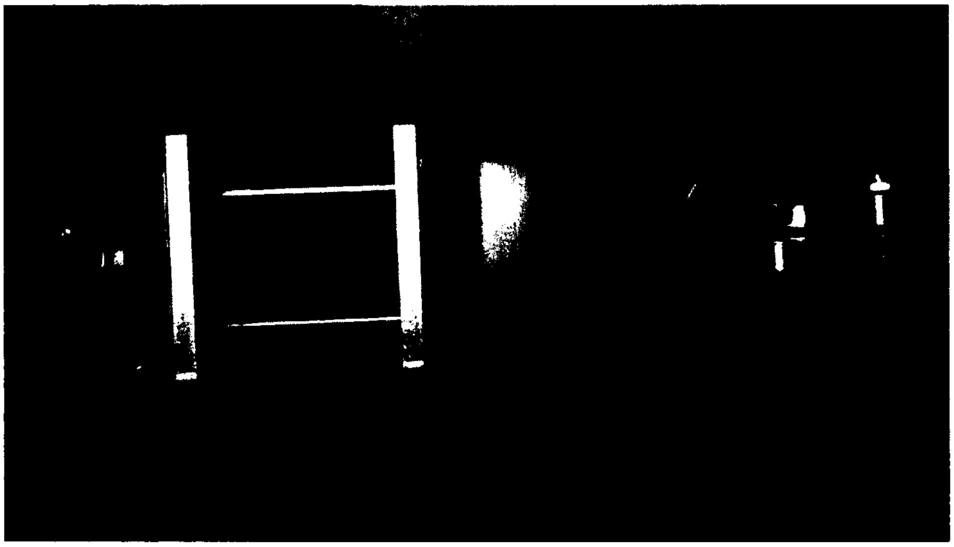


Figure A.5: Example of constructing a 2D component during a vice press, and 2D magnet placement tool (left) and 2D component (right) inside the vice.

A.3 3D Components

The materials used for constructing the 3D components include:

- Eden 333 Polyjet rapid prototyping machine
- Vero Gray resin
- Neodymium (NdFeB) disc magnets; 1/16" × 1/32" (diameter × height), grade N50
- Magnetic pole identifier
- Vice; built from a PanaVise bench clamp mount, a PanaVise low profile base, and a PanaVise low profile head
- Sharpie paint pens; oil based, fine point (colours: black, white, purple, yellow, green, and orange), and extra fine point (colours: red and blue)
- Rhino3D version 4.0 CAD software

The method for constructing the 3D components has the following eight steps:

1. Create the CAD files using Rhino3D for the 3D magnet placement tool (Figure A.6, page 231) and for the 3D components, based on the specifications in Figure A.7 (page 232) and in association with the 3D component shape space (to determine key, lock, and neutral information locations on each 3D component, as well as any 3D component interaction markers on neutral shapes adjacent to information locations).

2. Fabricate four 3D magnet placement tools (two with a magnet in the centre, and two with a magnet in the corner) and the 3D components using an Eden 333 Polyjet rapid prototyping machine with Vero Gray resin and the CAD files.

3. Insert three neodymium disc magnets in each 3D magnet placement tool, to create one tool with magnetic north polarity and the other with magnetic south polarity (identify polarity using the magnetic pole identifier).

4. Paint the 3D magnet placement tools, using the Sharpie paint pens (blue for magnetic north and red for magnetic south), to complete the construction of the 3D magnet placement tools.

5. Insert magnets into the 3D components by first identifying the appropriate 5-magnetic-bit pattern for each key and lock, and then placing magnets on the appropriate 3D magnet placement tool (two magnets for a key shape and three magnets for a lock shape) and using the vice to insert the magnets into the appropriate location in the 5-magnetic-bit pattern (Figure A.8, page 233).

6. Remove the 3D magnet placement tool and the 3D component from the vice, and separate the 3D component placement tool and the 3D component (the extra magnet will dislodge and create an air gap on the component where the magnets were inserted; this follows the 3D component physical encoding scheme for minimizing key-to-lock error interactions provided in Table 4.7 (page 101).

7. Repeat steps seven and eight until all magnets have been inserted into all the 3D components.

8. Paint, using the Sharpie paint pens, the magnets that have been placed in the components (blue for magnetic north and red for magnetic south) and any 3D component interaction markers (refer to Table 4.7, page 101, to use the appropriate colours), to complete the construction of the 3D components.

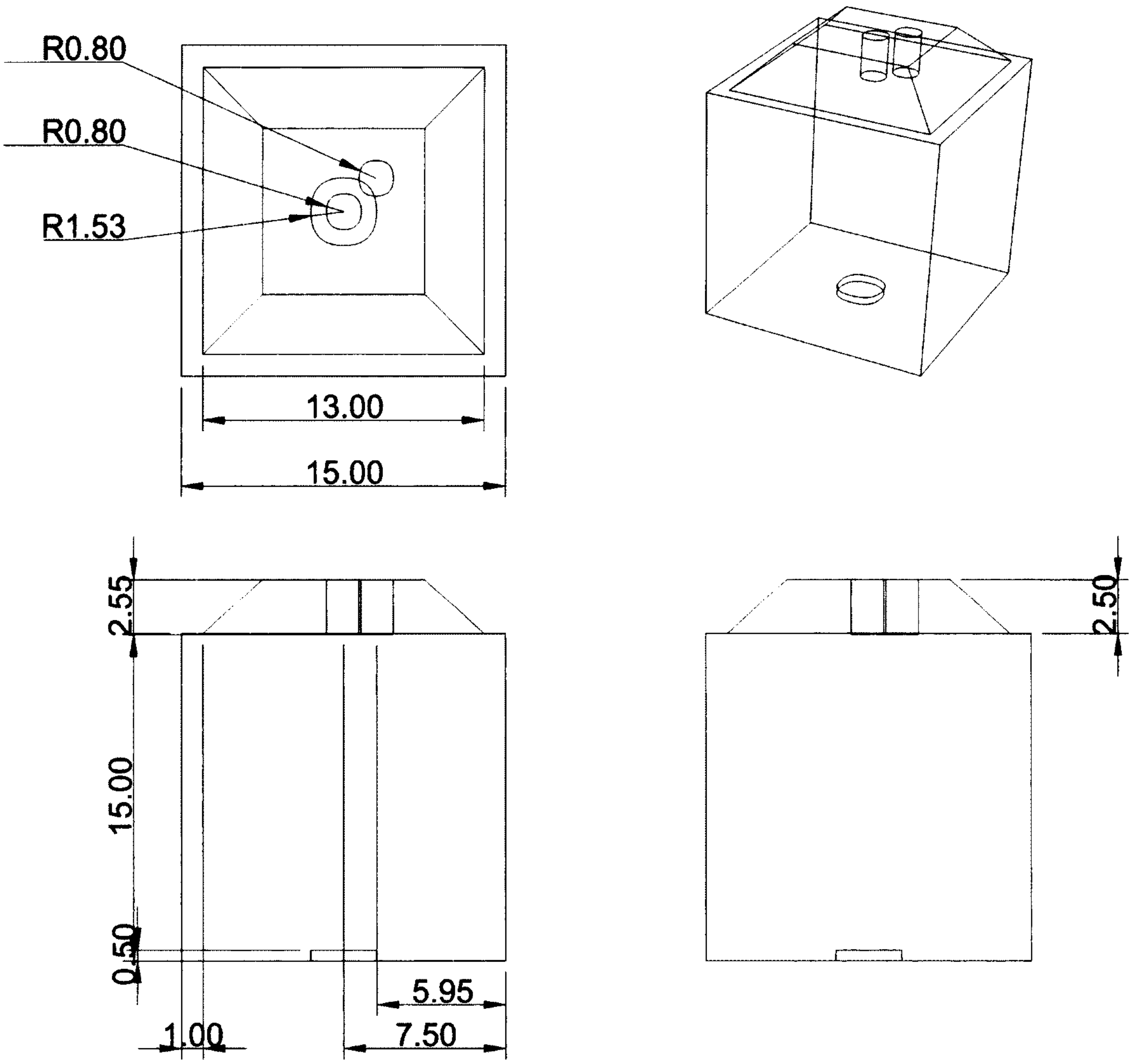


Figure A.6: 3D magnet placement tool specifications (counter-clockwise from top right: perspective, top, front, right views), with all construction units in millimetres; note two magnet locations are shown at the top of the component (centre and corner), however only one magnet location should be used per tool (two locations are show here to reduce the number of technical drawings).

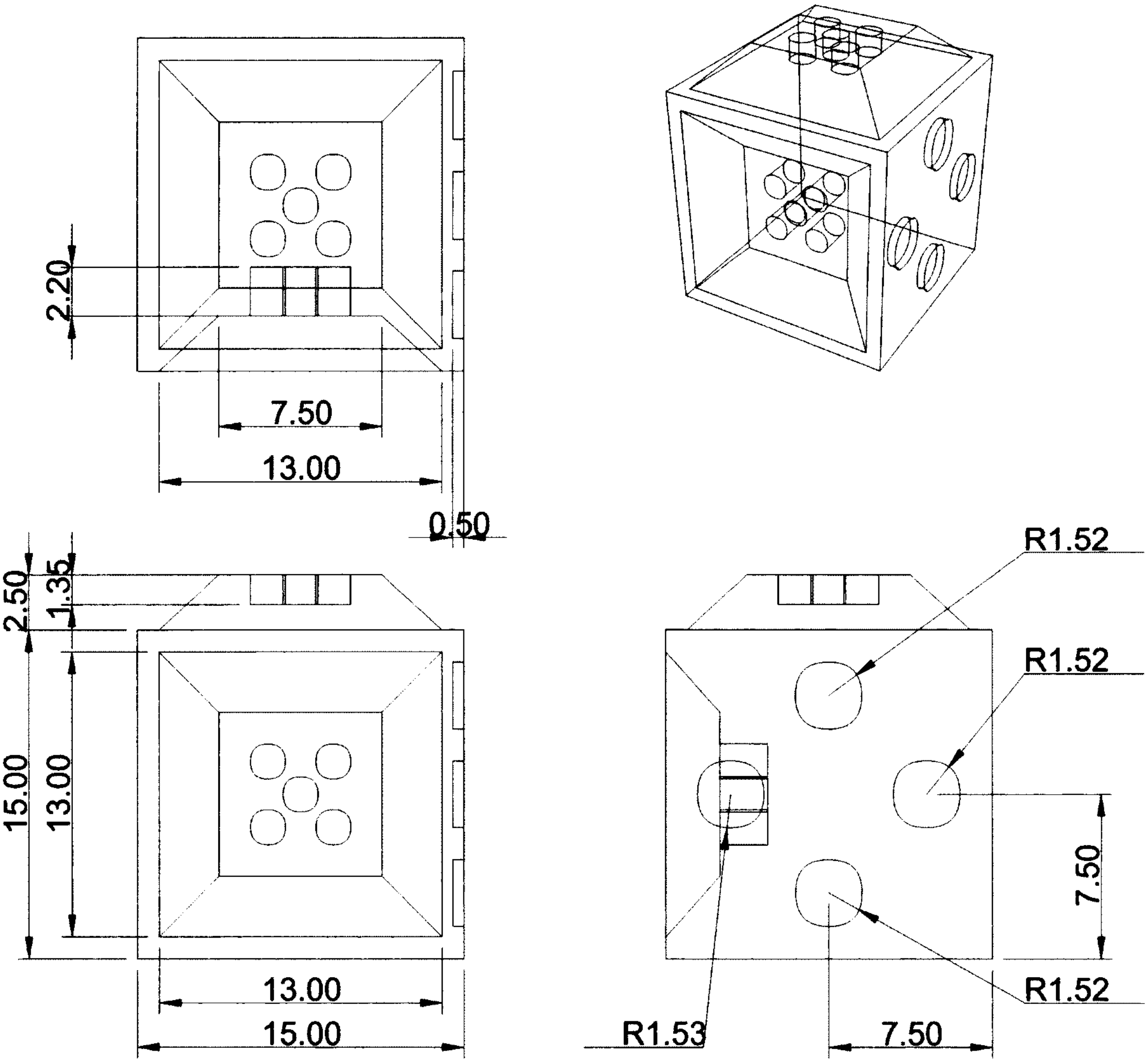


Figure A.7: 3D Component specifications showing the base component dimensions, key shape, lock shape, and interaction markers (counter-clockwise from top right: perspective, top, front, right views), with all construction units in millimetres.

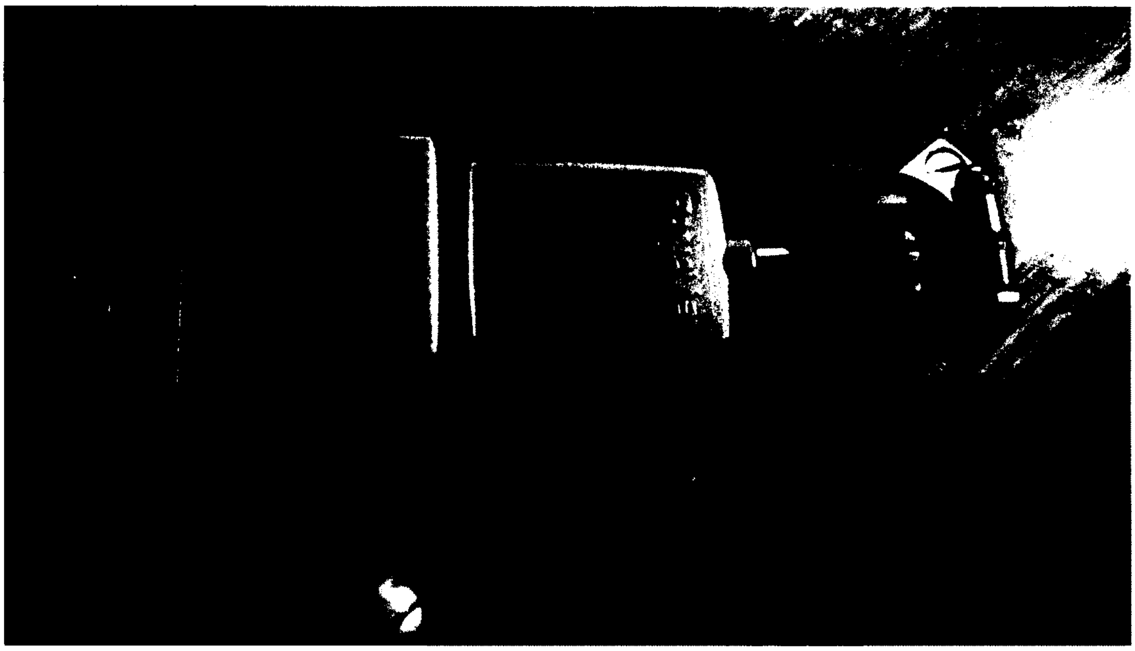


Figure A.8: Example of constructing a 3D component during a vice press, and 3D magnet placement tool (left) and 3D component (right) inside the vice.

Appendix B

Environment Specifications

The following is a list of the materials and methods used for constructing the physical environments corresponding to the original Penrose system (Penrose & Penrose 1957) presented in *Chapter 3: Physical Information Analysis*, and the 2D and 3D experiments presented in *Chapter 5: Programming Self-Assembling Systems*, *Chapter 6: Evolving Self-Assembly Rule Sets*, and *Chapter 7: Staging the Self-Assembly Process.*

B.1 Penrose Environment

The materials used for constructing the Penrose environment include:

- Clear acrylic sheet (3mm in thickness)
- Screws (5/8" in length, pan head, 6-32 UTS, 18-8 grade stainless steel)
- Wing nuts (6-32 UTS, 18-8 grade stainless steel)
- Trotec Speedy 300 Laser Engraver machine
- Adobe Illustrator Creative Suite 4

The method for constructing the Penrose environment has the following three steps:

1. Create the CAD files using Adobe Illustrator for the Penrose environment (Figure B.1).
2. Fabricate the Penrose environment parts (front, middle, and back) using a Trotec Speedy 300 Laser Engraver machine using the acrylic sheet and the CAD files.
3. Construct the Penrose environment using the screws and wing nuts (Figure B.2).

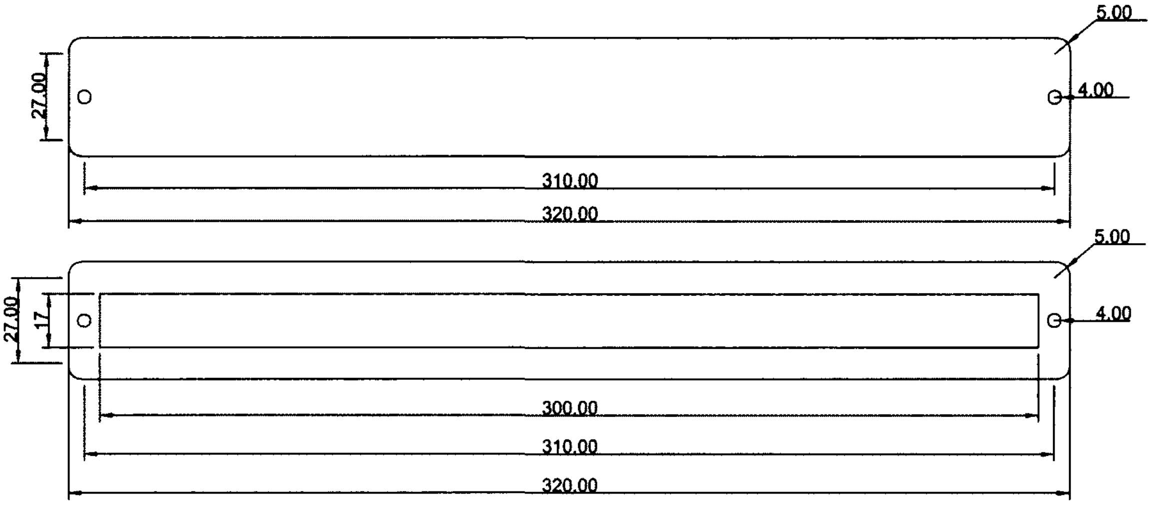


Figure B.1: Specification for the three parts, front and back parts (top) and centre part (bottom), making the Penrose environment (all construction units in millimetres; note hole diameter is 4 millimetres).

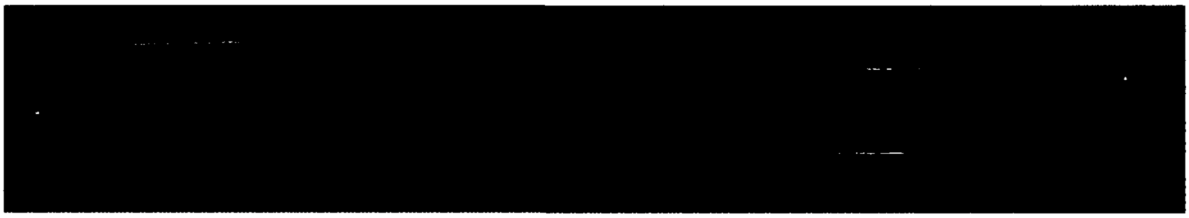


Figure B.2: Fully constructed physical Penrose environment.

B.2 2D Environment

The materials used for constructing the 2D environment include:

- Dimensions Elite rapid prototyping machine
- ABS plastic
- Trotec Speedy 300 Laser Engraver machine
- Acrylic sheet (2mm in height)
- Screws (1" in length, pan head, 6-32 UTS, polycarbonate)
- Hex nuts (6-32 UTS, polycarbonate)
- Wing nuts (6-32 UTS, polycarbonate)
- Max Mix II Vortex Mixer
- Rhino3D version 4.0 CAD Software
- Adobe Illustrator Creative Suite 4

The method for constructing the 2D environment has the following nine steps:

1. Create the CAD files for the using Rhino3D for the tray base (Figure B.3) and the tray mounting bracket (Figure B.4 and Figure B.5, page 238).

2. Fabricate the tray base and the tray mounting bracket using a Dimensions Elite rapid prototyping machine with the sparse-fill option (used when printing to save material, as well as produce a rough textured surface), using ABS plastic and the CAD files.

3. Create the CAD files for the tray lid using Adobe Illustrator (Figure B.6, page 239).

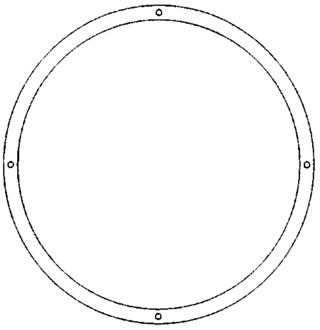


Figure B.3: Tray base specification (top view) - the outer radius is 135; the inner radius is 125; the outer wall height is 9; the inner wall height is 6; the radius for each screw hole is 2 (all construction units in millimetres).

4. Fabricate the tray lid using a Trotec Speedy 300 Laser Engraver machine, using acrylic sheet.

5. Place four screws in the appropriate holes on the tray mounting bracket and secure with hex nuts (screw heads are at the bottom of the tray mounting bracket and hex nuts are at the top, Figure B.7, page 239).

6. Secure the tray mounting bracket to the Maxi Mix II Vortex Mixer using the two screws supplied with the mixer (Figure B.7, page 239).

7. Slide the tray base over the screw ends on the tray mounting bracket until it is flush with the hex nuts (Figure B.7, page 239).

8. Place the tray lid over the screw ends on the tray mounting bracket until it is flush with the tray base, and secure the tray lid to the tray base using four wing nuts (Figure B.7, page 239).

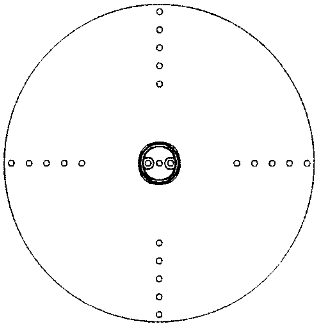


Figure B.4: Tray mounting bracket specification (surface specification, top view) - the outer radius is 220; in each group of five screw holes, the distance from closest to furthest (centre of the hole to the centre of the part) is 55, 67.5, 80, 92.5, 105; the radius for each screw holes is 2 (all construction units in millimetres).

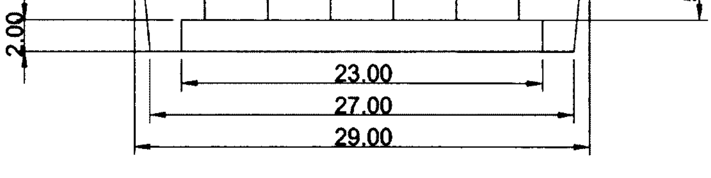


Figure B.5: Tray mounting bracket specification (mount detail, front view) - smaller bracket surface for clarity (all construction units in millimetres).

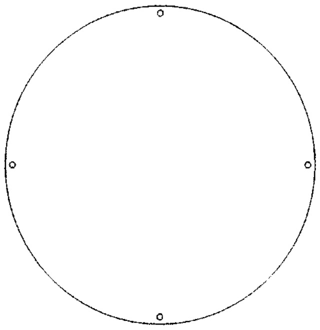


Figure B.6: Tray lid specification (top view) - the radius is 135; the height is 2; the radius for each screw hole is 2 (all construction units in millimetres).

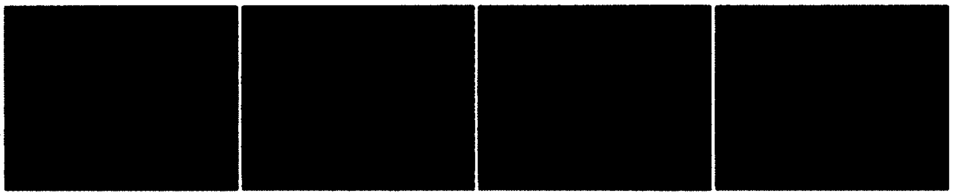


Figure B.7: From left to right: steps five to eight to construct the 2D environment.

B.3 3D Environment

The materials used for constructing the 3D environment include:

- Trotec Speedy 300 Laser Engraver machine
- Acrylic sheet; transparent, 3mm in height
- Mill board; 0.8 mm in height
- Screws; first type (5/8" in length, pan head, 6-32 UTS, 18-8 grade stainless steel), second type (1/4" in length, pan head, 6-32 UTS, 18-8 grade stainless steel)
- Hex nuts; 6-32 UTS, 18-8 grade stainless steel
- Glue; Loctite, regular, gel

- Angle brackets; 1, stainless steel
- New Brunswick Scientific Excella E1 Platform Shaker
- Adobe Illustrator Creative Suite 4

The method for constructing the 3D environment has the following six steps:

1. Create the CAD files using Adobe Illustrator for the jar rack parts (Figure B.8).

2. Fabricate the jar rack parts (two bottom parts X and one bottom part Y, two side parts, and one top part) using a Trotec Speedy 300 Laser Engraver machine using mill board for the jar sleeve and acrylic for the remaining environment jar rack parts, and the CAD files.

3. Glue the three bottom jar rack parts together (bottom parts X both below bottom part Y).

4. Construct the jar rack by using the screws, hex nuts, and corner braces to secure the bottom, sides, and top parts of the jar rack, and using the screws and hex nuts to secure the jar sleeve to the top of the jar rack.

5. Fold the overhang pieces of the jar rack sleeve over the holes for the jars in the top par of the jar rack.

6. Place the jar rack on the shaker (Figure B.9, page 242), and secure the jar rack to the shaker using the screw on the side of shakers platform (these screws are supplied with the shaker), to complete construction of the environment.

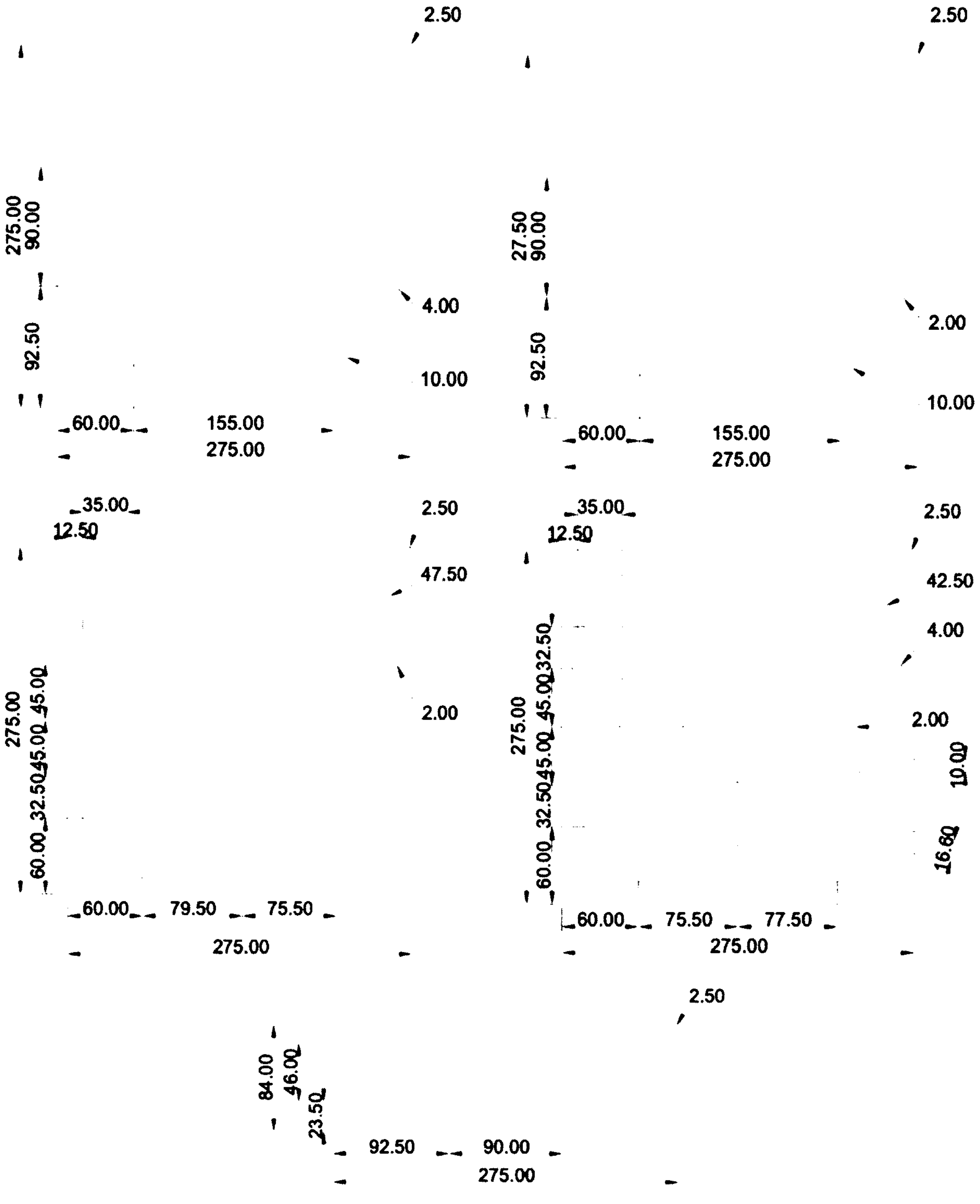


Figure B.8: Environment parts for jar rack (from top left to bottom: bottom part X, bottom part Y, top part, jar sleeve, and side part), with all construction units in millimetres.

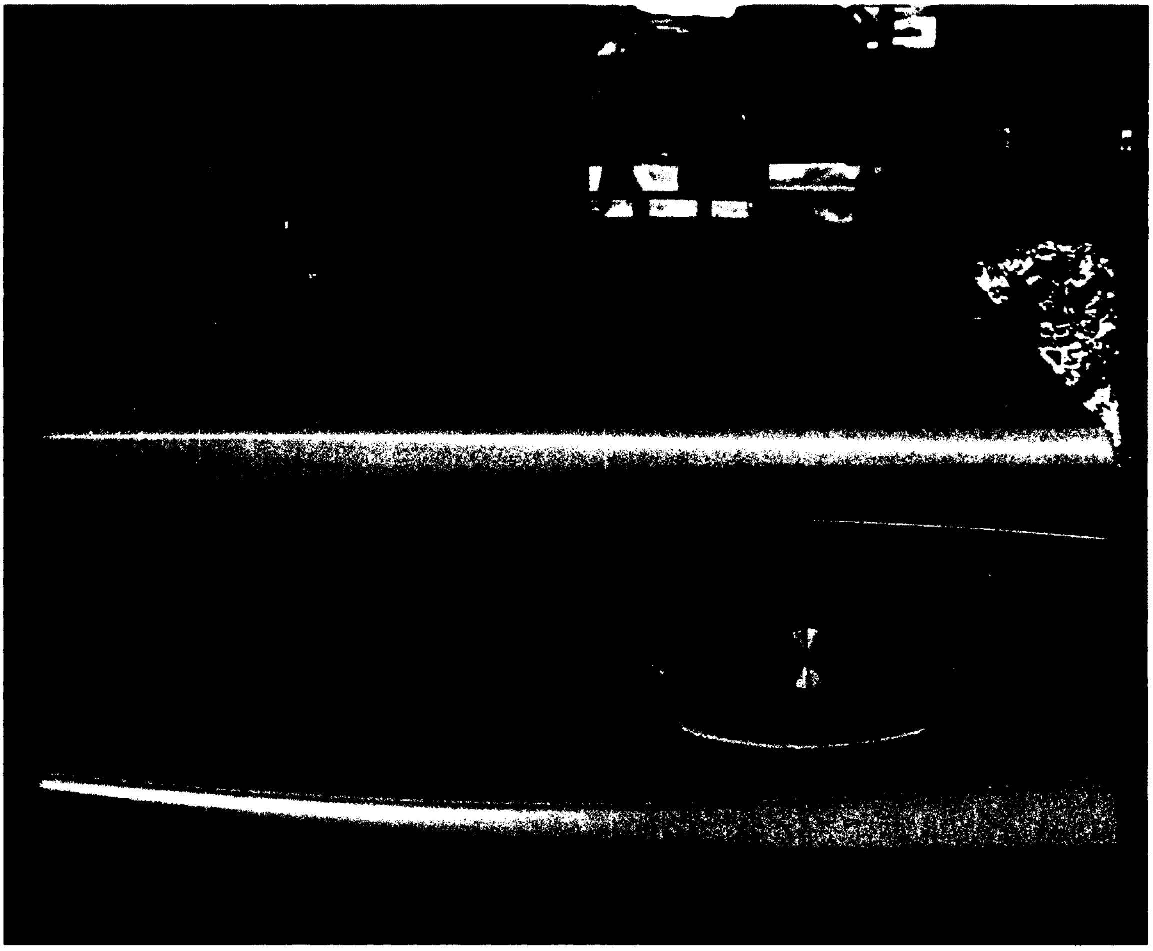


Figure B.9: Photograph of the 3D environment with the jar rack secured to the shaker, and a jar with mineral oil containing components as an example (jars of mineral oil are used to finalize the 3D components environment; details are provided in *section C.3: Experimental Procedures for the 3D Systems*).

Appendix C

Experimental Procedures

The following is a list of the experimental procedures corresponding to the original Penrose system (Penrose & Penrose 1957) presented in *Chapter 3: Physical Information Analysis*, and the 2D and 3D experiments presented in *Chapter 5: Programming Self-Assembling Systems*, *Chapter 6: Evolving Self-Assembly Rule Sets*, and *Chapter 7: Staging the Self-Assembly Process.*

C.1 Experimental Procedure for the Penrose Systems

The experimental procedure for the Penrose experiments has the following six steps:

1. Place the middle Penrose environment part on top of the back Penrose environment part.

2. Place the α and β parts, and the corresponding seed complex parts on the back Penrose environment part according to the initial configuration.

3. Place the top Penrose environment part on top of the middle Penrose environment part.

4. Secure the Penrose environment parts together using the screws and wing nuts.

5. Shake the Penrsoe environment (1D) by hand six times (three times to the left, and three time to the right) with enough force to ensure that all the components slide to one end of the Penrose environment.

6. Record the state of the system.

C.2 Experimental Procedure for the 2D Systems

The materials required to conduct the physical 2D experimental procedure include:

- Stopwatch
- 3" c-clamp
- Hex nuts (6-32 UTS, 18-8 grade stainless steel)

The experimental procedure for the 2D experiments has the following seven steps:

1. Set the continuous speed control on the Maxi Mix II Vortex mixer to 1,050 rpm.
2. Secure the mixer to a table, using a 3" c-clamp and six hex nuts (to help secure the c-clamp to the back of the mixer).
3. Randomly place the 2D components on the surface of the tray (trying to ensure that complementary bonding sites on the 2D components are not in-line with each other).
4. Secure the tray lid.
5. Turn the mixer on, and start the stopwatch.
6. Run the mixer for 20 minutes (for the 2DP experiments, the 2DE experiments, or the non-staged 2DS experiments), or for two 10 minutes intervals (for the staged 2DS experiment, and randomly place the 2D components corresponding to the second time interval after the first time interval has completed).
7. Record the state of the system, observations including: the number of target structures created, the number of matching errors (between conflicting physical information, where no fits rule is applicable), and the number of assembly errors (partial attachment where a fits rule is applicable).

C.3 Experimental Procedure for the 3D Systems

The materials required to conduct the physical 3D experimental procedure include:

- Graduated cylinder
- Mineral oil; Rogier Pharma light grade
- Stopwatch
- Jars; VWR clear glass wide mouth, plastic lid with rubber liner, 500 mL capacity, 91 mm × 95 mm (diameter × height)

The experimental procedure for the 3D experiments has the following six steps:

1. Use the graduated cylinder to measure 325 mL of mineral oil for each jar used: 3DP (5 jars), 3DE (3 jars), or 3DS (3 jars).
2. Place the jars of mineral oil in the jar rack (Figure C.1).
3. Randomly place the 3D components for each experiment into the appropriate jar, and secure the jar lid.
4. Turn the shaker on by setting the speed to 32.5 rpm, and start the stopwatch.
5. Run the shaker for 20 minutes (for the 3DP and 3DE experiments), for 40 minutes (for the non-staged 3DS experiments), or for two 20 minute intervals (for the staged 3DS experiments, and randomly place the 3D components corresponding to the second time interval after the first time interval has completed)
6. Record the state of each system, observations including: the number of target structures created, the number of matching errors (between conflicting physical

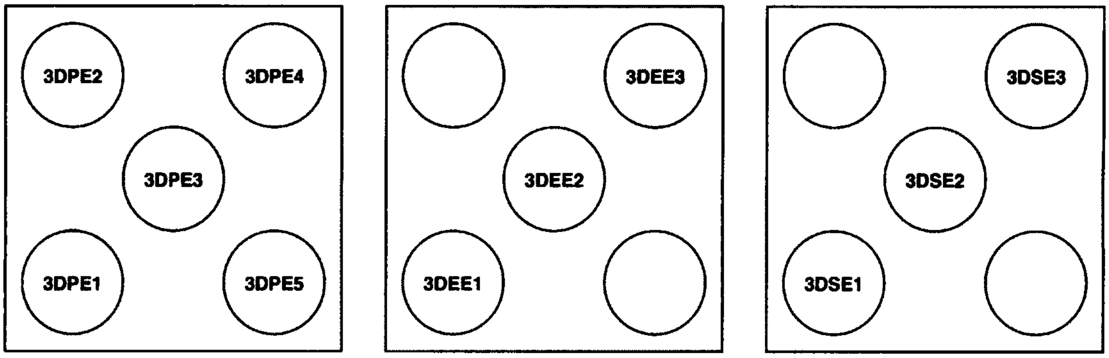


Figure C.1: Jar configurations (top view) for the 3DP experiments (left, 3DPE1 - 3DPE5), the 3DE experiments (centre, 3DEE1 - 3DEE3), and the 3DS experiments (right, 3DSE1 - 3DSE3).

information, where no fits rule is applicable), the number of rotation errors (between complementary components), and the number of assembly errors (partial attachment where a fits rule is applicable).