

Self-organizing Computation

A Framework for Generative Approaches to Architectural Design

by

Taro Narahara

Master of Science in Architectural Studies, Massachusetts Institute of Technology, 2007

Master of Architecture, Washington University in St. Louis, 1997

Bachelor of Science in Mathematics, Waseda University, Tokyo, Japan, 1994

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Design

At the Harvard University Graduate School of Design

September 2010

Copyright ©2010 by Taro Narahara
All rights reserved

Thesis Committee Chairman:

Martin Bechthold
Professor of Architectural Technology

Self-organizing Computation

A Framework for Generative Approaches to Architectural Design

by

Taro Narahara

Master of Science in Architectural Studies, Massachusetts Institute of Technology, 2007

Master of Architecture, Washington University in St. Louis, 1997

Bachelor of Science in Mathematics, Waseda University, Tokyo, Japan, 1994

Submitted in partial fulfillment of the requirements
for the degree of
Doctor of Design
at the Harvard University Graduate School of Design

September 2010

Copyright © 2010 by Taro Narahara

The author hereby grants Harvard University permission to reproduce and distribute copies of this thesis document, in whole or in part, for educational purposes.

Signature of the Author.....

Taro Narahara

Harvard University Graduate School of Design

Certified by.....

Martin Bechthold

Professor of Architectural Technology

Thesis Committee, Chairman

Accepted by.....

Antoine Picon

Professor of the History of Architecture and Technology

Director, Doctor of Design Program

Harvard University Graduate School of Design

Abstract

This thesis proposes a conceptual framework for applications of self-organizing logics in generative design systems. The methods introduced in this thesis are in an abstract and conceptual form that explores one possible future direction of computational design strategy. In order to explain the potential of this problem-solving direction, general aspects of what our contemporary practice in architecture and urban design is facing will be discussed in response to the increasing complexity in our culture. However, the main focus of this thesis is not on providing immediate solution methods to resolve any specific professional problems in contemporary architecture. Rather, the thesis investigates the emergent characteristics of this method that can potentially evolve new design solutions over time, and shows how tools employing the method can be used for design collaboration with humans, rather than simply as passive evaluation and analysis tools. The thesis foresees important potential for this new design direction inspired by self-organizing computation (SOC) and speculates regarding its potential areas of application in architecture and urban design.

In recent years, many scientists have started to gain the advantages of self-organizing systems in nature through their computational models in areas such as telecommunication networks and robotics. Main advantages of such systems are robustness, flexibility, adaptability, concurrency, and distributedness.

One of the unique characteristics of SOC is its non-reliance on any external knowledge. As with many conventional computational methods in architecture, imposing existing design patterns or transformation sequences is beneficial when one wants to efficiently derive what appear to be the subjects of our recognitions. However, reliance on a pre-existing template might preclude the possibility of discovering what original inputs naturally turn into.

SOC is a computational approach that brings out the strengths of the dynamic mechanisms of self-organizing systems: structures appear at the global level of a system from interactions among its lower-level components. In order to computationally implement the mechanisms, the system's constituent units (subunits) and the rules that define their interactions (behaviors) need to be described. The system expects emergence of global-scale spatial structures from the locally defined interactions of its own components.

Acknowledgments

I would like to sincerely thank Professor Martin Bechthold for his valuable comments and feedback. Through his critical guidance, he has been helping me build an academic foundation in the field of architectural research. Without his persistent help this dissertation would not have been possible. I also thank him for the opportunity to work on robotic fabrication as a research fellow for two years.

Without meeting Professor Kostas Terzidis, I could never have reached where I am now. His unique philosophical insights in computation, architecture, and complexity theory will continue to inspire me. I am deeply grateful for his continuous encouragement, guidance, and support during the research process.

I would also like to thank Professor Takehiko Nagakura – a man with a rare ability to integrate practice and academics – for his insightful guidance and constant support during the development of this thesis. He has been my academic adviser since I first came to MIT, and has been a superb mentor and educator.

I would also like to extend my sincerest thanks to President Ito at Forum8 Co. Ltd. in Tokyo, Japan, and to Dr. Kobayashi Yoshihiro for leading the virtual reality research group, and for their constant support for my interests in agent-based simulation.

I am also indebted to Professor Ingeborg Rocker at the Graduate School of Design for giving me the opportunity to teach at GSD. Teaching and running Rhinoscript and processing workshops added another exciting element to my Harvard life.

I would like to thank Dr. Robert A. Irwin at MIT for his guidance on academic writing and for his suggestions about readings on the topics of emergence and self-organization.

I want to give a special mention to Professor Larry Kubota for his continuous encouragement, guidance, and support during the research process; to Professor Shun Kanda for the opportunity to assist his advanced design studio at MIT; and to my former employers Professor Hiroto Kobayashi and Richard Gluckman. I am also grateful to Professor Adrian Luchini, my mentor as an architect, and to my colleagues at Harvard and MIT: Katsunobu Sasanuma, Kenfield Griffith, Jae Wan Park, and Shouhei Matsukawa.

Finally, I would like to sincerely thank my parents for their limitless support and understanding. I have respected my father's extremely diligent attitude toward his academic research, and have followed his path. I admire my mother's strength for supporting my family when we were weak. I would also like to thank my grandmother, who passed away this March. She was one year short of one hundred years old, and I could never beat her at Reversi or any puzzles. I would like to dedicate this dissertation to my father, who passed away last year.

The Committee Members for this thesis are

.....
Martin Bechthold

Professor of Architectural Technology
Thesis Committee, Chairman
Harvard University Graduate School of Design

.....
Kostas Terzidis

Associate Professor of Architecture
Thesis Committee, Adviser
Harvard University Graduate School of Design

.....
Takehiko Nagakura

Associate Professor of Design and Computation
Thesis Committee, Adviser
Massachusetts Institute of Technology

To my parents,

and in memory of

my father,

Yoshiyuki Narahara,

and my grandmother,

Chika Hiroe

Table of Contents

Abstract	iii
Acknowledgements	iv
1 Introduction	1
1.1 Space + Time	2
1.2 Complexity, Time, and Adaptation	2
1.3 Technological Advancement	5
1.5 Definitions	6
1.5 Computation Systems	9
1.6 Hypothesis	10
1.4 Outline of Chapters	12
2 General Characteristics of Self-organizing Strategies	13
2.1 Natural Systems and Their Computational Simulations	14
2.1.1 Introduction	14
2.1.2 Self-organization	15
2.1.3 Synthetic versus Natural Systems	19
2.1.4 Collective Construction	21
2.1.5 Applications of Self-organization	29
2.2 Artificial Systems	35
2.2.1 Reframing Human Design Activities	35
2.2.2 Physical Implementations	42
2.2.3 Metabolist Movement - Scale and Size of Subunits	42
2.2.4 Process Planning by Isozaki	44
2.2.5 Construction Automation during the 1980s in Japan	45
2.2.6 Swarm Intelligence: Self-reconfigurable Robots	49
2.2.7 Self-Replicating Machines: 3D Printers	53
2.3 Urban-Scale Systems: City Formation and Emergent Growth	56
2.3.1 Kowloon Walled City – Architecture in a State of Anarchy	56
2.3.2 Participatory Design Guided by Professionals	61
Review of Urban Simulations: Computational Methods	65
2.3.3 Fractals, DLA, and Agents: Examples from M. Batty at CASA	65
2.3.4 Shape Grammar and L-System: City Engine	67
2.3.5 Superimposition Technique by C. Alexander and M. L. Manheim	72

2.4	Summary and Analysis of the Background	76
2.4.1	Subunits	77
2.4.2	Scale	78
2.4.3	Time	80
2.4.4	Discussion and Critique	81
2.4.5	Conclusions	84
3:	Computational Methods of Self-organization	86
3.1	Computational Methods: Three Phases of Computational Applications in Design	87
3.1.1	Method 1: Evaluation	88
	- Quantitative / Qualitative Evaluation	88
3.1.2	Method 2: Design Search	90
	- Deterministic / Non-deterministic Search	92
	- Genetic Algorithm (GA)	93
	- Traveling Salesman's Problem (TSP)	94
	- Multi-objective Optimization Problems (MOP)	96
	Plain Aggregation Methods (Weighted-sum)	97
	Population-based Non-Pareto Approaches	97
	Pareto-based Approaches	98
	- Qualitative Evaluations	99
	Interactive GA	100
	Feedback from Physical Experiments	101
	- Projecting an Arrow from Method1 to Method2	101
	Bayes' Theorem	102
	From Pedestrian Simulation to Topology Optimization	102
	Flipping the Arrow	104
3.1.3	Method 3: Growth + Adaptation	106
	- Fibonacci Sequence	107
	- Evolutionary Game	108
	- Examples:	110
	A. Conventional Renovation Scenarios	110
	B. Modular System (Kit-of-Parts)	110
	C. Self-organizing Growth	111
3.2	Self-organizing Computation	113
3.2.1	Self-organizing Computation	113
3.2.2	Key Attributes of Self-organization	113
3.2.3	Key Properties of Self-organizing Phenomena	115
3.2.4	Emergent Formations: Examples of Self-organizing Computation	116
3.2.5	Lane Formation	117
3.2.6	Circle Packing	124
3.3	Concluding Note: Trans-dimensional Topology Concept	128

4	Experiments in Evaluation and Search (Design Search)	131
4.1	Evaluation: Implementing Physical Reactions to CAD System	132
4.1.1	Elastic Spring Mass Object	133
4.1.2	Finite Difference Method	137
4.1.3	Stress Display	138
4.1.4	Kinetic Objects	141
4.2	Design Search: Turtle Implementation of L-system	143
4.2.1	Introduction	143
4.2.2	Method	143
4.2.3	Design Generator (Building Engine)	144
4.2.4	Generative Grammar Instructions	146
4.2.5	Evaluation Sequence Using Evolutionary Computation	147
4.2.6	Auto-Generation of Design Patterns	150
4.2.7	Fitness Functions: Evaluations	151
4.2.8	Evolutionary Runs	155
4.2.9	Procedural Representation Using Parametric L-System	158
4.2.10	Future Works	160
5	Experiments in Growth and Adaptation	161
5.1	Growth: Diffusion-limited Aggregation (DLA)	162
5.1.1	Laplacian DLA Model Based on a Probability	163
5.1.2	Architectural Applications of DLA	165
5.1.3	Conclusion for DLA Experiments	176
5.2	Adaptation: Physical implementation	178
5.2.1	Distributed Systems	179
5.2.2	The Nelder-Mead Method: Physical Implementation	183
5.2.3	Discussion and Critique	189
5.2.4	Conclusion	191
6	Application of Self-organizing Computation: From Prediction to Synthesis	193
6.1	Objectives	197
6.2	Emergent Design System	199
6.2.1	Development of a Design System: Technical Note	199
6.2.2	Environment	199
	- Terrain Generation	199
	- Importing Surfaces	200
	- Random Midpoint Displacement Method	201
	- Noise Functions:	202

6.2.3	Agents as Wandering Settlers:	203
	- Time Scale:	205
	- Attraction to Slope:	206
	- Slope Definition by Agents:	206
	- Traffic Intensity: Chemical Trail	208
	- Chemical-reduction Rate (Decay Rate)	210
	- Diffusion Rate	211
	- Attraction to Destinations:	211
	- Direct Path Systems:	212
6.2.4	T + D Model	214
	- Minimal Way Systems	216
6.2.5	S + T + D Model	219
	- Summary of Results	231
6.2.6	Emergent Behaviors of Agents	232
	- Early Periods	232
	- City Emergence	236
	- Itinerary for Agents	237
	- Building Behaviors	242
	- Negotiation between Agents and Buildings	245
6.2.7	Experiments	247
	- Results	250
6.3	Conclusion	261
6.3.1	Degree of Reliance on External Knowledge	261
6.3.2	Scale of Space and Time	262
6.3.3	Limitations	263
6.3.4	Future Work	266
7	Conclusions	269
7.1	Two Difficulties in Computational Approaches to Generative Design in Architecture	271
7.2	Two Common Approaches	271
	- Reduce Possibilities and Choose among a Smaller Subset	272
	- Employ Methods That Feature Self-directed Solution-seeking Behaviors	273
7.3	Summary: Comparison of Proposed Systems	275
7.4	Application Areas That Benefit from the Advantages of Self-organizing Computation	
	- Growth Model	284
	- Performance-based Design	285
	- Objectives in Transition	285
	- Decentralized Systems and System Control	286

7.5	Remarks and Implications	288
	- Explicit or Gradient Representations	288
	- Degree of Reliance on External Knowledge	289
	- Induction Methods	290
	- Modeling and Designing (Simulation and Generation)	292
	- Future Research	293
	- A Question about Application Scales and Areas	293
	Appendix:	296
	List of Figures	
	References	297
	Relevant Publications by the Author	305

Chapter 1

Introduction

This thesis proposes a conceptual framework for applications of self-organizing logics in generative design systems. The methods introduced in this thesis are in an abstract and conceptual form that explores one possible future direction of computational design strategy. In order to explain the potential of this problem-solving direction, general aspects of what our contemporary practice in architecture and urban design is facing will be discussed in response to the increasing complexity in our culture. However, the main focus of this thesis is not on providing immediate solution methods to resolve any specific professional problems in contemporary architecture. Rather, the thesis investigates the emergent characteristics of this method that can potentially evolve new design solutions over time, and shows how tools employing the method can be used for design collaboration with humans, rather than simply as passive evaluation and analysis tools. The thesis foresees important potential for this new design direction inspired by self-organizing computation and speculates regarding its potential areas of application in architecture and urban design.

1.1 Space + Time

In today's design methodologies, we normally try to anticipate all current and future design requirements and potential changes for buildings prior to construction and endeavor to resolve all issues in a single static solution. However, this static solution may not always be able to respond to ongoing radical population growth and environmental changes. Moreover, the physical scale of buildings and the complexity involved in building programs have been increasing to unprecedented levels. In such conditions, increased use of process-based four-dimensional design strategies (space + time) can be anticipated.

This four-dimensional design thinking is not only promising for developing more flexible and adaptable architecture, but also for expanding the territory of design to systems issues. The thesis investigates the potential of architecture and urban design being a system that can grow over time and argues that the computational strategies inspired by self-organizing logics are a promising direction for establishing such systems. The neglected feasibility and application area of extensible systems in architecture will be reviewed, and the thesis speculates as to the possibilities of open frameworks for design using computational methods.

1.2 Complexity, Time, and Adaptation

One of the main concerns in architecture today is the increasing quantity of information to be processed during design and the level of complexity involved in most building projects. As globalization and economic development increase, large-scale urban

development has become ever more essential. Complex threads of relationships among buildings and urban infrastructures are intertwined to produce inseparable connections. Dependencies among these structures are extremely intense, not only pragmatically but also aesthetically. Nearly half a century ago, Christopher Alexander (1964) already foresaw these conditions and stated the following:

In any case, the culture that once was slow-moving, and allowed ample time for adaptation, cannot keep up with it. No sooner is adjustment of one kind begun than the culture takes a further turn and forces the adjustment in a new direction. No adjustment is ever finished. And the essential condition on the process – that it should in fact have time to reach its equilibrium – is violated.

Nowadays, most buildings and infrastructure designs require dynamic and collaborative engagements by multiple professionals, and a conventional knowledge-based approach alone may not be able to respond to emerging building types. For example, housing projects for thousands of people have been emerging in urban areas, and demands for planning and design of buildings with multiple occupancies and complex programs are becoming a challenge. The social impact of such buildings can completely alter the behavioral dynamics and physical conditions of local environments, and often lead to redefinitions of transportation systems and infrastructures at far greater scales. Moreover, lack of flexibility and adaptability to ever-changing environments has resulted in the need to carry out expensive and invasive operations of demolition and reconstruction.

Conventional solutions for multifunctional large-scale complexes, high-rise office towers, and housing complexes are often to design a plan resolving all the problems within a

single floor and simply stack one on top of another (Figure 1.1). This approach may satisfy a quantity of initial requirements on a temporary basis; however, there are always other potential spatial configurations worthy of investigation. Selecting optimal solutions that can accommodate unpredictable additions and renovations for the future adaptation has become a difficult task for architects. Our ultimate goals for successful design have shifted to seeking solutions for satisfying more long-term needs in a flexible manner. In response to the increasing complexity in our culture, pioneering works using computational design methods are reviewed. Advantageous use of computation to resolve these complexities is reviewed and advanced in this thesis.



Figure 1.1 – In these high-rise residential towers in Hong Kong, housing thousands of people, identical floors are simply stacked one on top of another.

There have been some efforts to create physical architectural systems that can actively accommodate future changes of their environments and emerging new objectives. Development of flexible and adaptable architecture has been a perennial theme among practitioners, and some of the practical limitations seen among the attempts by Metabolists in the 1960s clearly indicate the difficulties of designing universal subunits that could endlessly tolerate technological, environmental, and circumstantial changes associated with structures. In such cases, goals and objectives of buildings are also often dynamic properties of the structures, and design systems for such structures may require abilities to simulate and foresee unknown objectives from initial sets of essential conditions. This thesis investigates new possibilities of developing physical architectural systems with regard to current technological developments.

1.4 Technological Advancement

In addition to the emerging complexities in our building programs, recent advances in engineering have allowed architects to envision building structures as active responsive and performative units on far greater scales, and have started to introduce additional layers of complexity into building design. For example, new construction materials such as ultra-high-strength concrete can span far greater lengths and expand architectural scope and possibilities. Moreover, recent advancements in sensor technologies have opened possibilities for buildings to have active adaptable mechanisms.

These technical innovations are widening our solutions to current architectural problems; however, our current design strategies are not capable of fully utilizing these rapidly

expanding technological possibilities in architecture and urban development. Linear summations of knowledge-based conventional design strategies alone may not provide optimal solutions from widening spatial design repertoires, and new types of design strategies are anticipated.

1.5 Definitions

In architectural design, some building types require fewer changes in the future, but some require more. Besides seeking to minimize the need for future alterations or to economize on projected future costs of preventable maintenance work, the design methods aim, from the outset of design processes, at producing completed buildings that can tolerate as many future conditions as possible. Usually these processes are conducted by a professional (architect) as a leader, and the design directions are provided through comprehensive blueprints. In architecture, design and construction processes that satisfy conventional problems are empirically known to some extent as building typology and established construction methods. I will call this type of design approach “top-down.” In this design process, after a series of refinements through various design phases, physical construction usually follows based on the blueprints established at the earlier stages of design; therefore the system has the least tolerance for spontaneous changes from lower-level components during or after the construction. There will be fewer dynamic changes. This fact makes our construction process vulnerable to any type of change, such as changes in physical environments, numbers of occupancies, or types of specific uses. In other words, almost every possible scenario relating to the uses of the

buildings has to be anticipated and analyzed in advance to fulfill all kinds of consequential future requirements.

On the other hand, some buildings do not include a comprehensive solution for all the potential scenarios of the future from the outset. Instead, some of those buildings possess systems that allow them to adapt to future changes over time by altering their designs spontaneously based on simultaneous feedback from a number of simple entities (or agents) inside the system. These feedback systems can be effectively distributed to formulate globally satisfactory working solutions as a collective result. These methods do not always guarantee the best solution in a deterministic sense; however, they may prove effective where there is no deterministic and analytical means to derive solutions. As a natural consequence of adapting to radical population growth, sometimes these characteristics can be seen in low-cost housing developments in less regulated zones with no supervision by professionals. These developments from human designs, which will be reviewed in the next chapter, do not provide completely positive results; however, characteristics of dynamic adaptations seen in these examples suggest ideas for future computational implementations. I will call this type of design approach “*bottom-up*.”

Distributed control is one technical strategy to realize a feedback system inside a *bottom-up* system, and this strategy can be applied to control not only of a single structure but also of multiple structures. Inputs for this feedback system are fed from separated nodes and can be triggered by participation of independently acting agents with some intelligence. The entire system’s behavior is a result of feedback from multiple intelligent

sources, and this system is often called “collective intelligence.” Participatory design is one application of this concept through human group participation.

Self-organization is a characteristic that can be found in systems of many natural organisms: flocking of birds, collective building behaviors by various social animals and insects, pigmentations of cells in animal skin patterns, formation of dunes by sand particles, and so on. These behaviors are often called emergent behaviors, and **emergence** refers to “the way complex systems and patterns arise out of a multiplicity of relatively simple interactions,” according to Camazine et al. (2002). Original theories of self-organization, developed in the context of physics and chemistry, are defined as the emergence of macroscopic patterns out of processes and interactions defined at the microscopic level (Nicolis and Prigogine, 1977; Haken, 1983). Such systems’ behaviors display many characteristics that are similar to the aforementioned *bottom-up* approach in some artificial systems.

Not only in natural systems have we witnessed self-organizing growth processes, but also in some artificial systems. Beyond the scale of buildings, we have witnessed self-organizing growth processes in many formations of cities on large scales over long spans of time. Although results of these processes are not always successful in all aspects of design, the processes display heuristic and almost trial-and-error types of approaches that are robust and flexible enough to dynamically adapt to ever-changing environments. As the cities deliberately created by designers and planners rarely display the level of flexibility seen in these spontaneous city growth patterns, it is worth investigating their characteristics. Rigorous computational reinterpretation and application of the underlying

principles behind artificial self-organizing phenomena will possibly enhance and elaborate the advantages of these systems up to a more practical level.

1.6 Computation Systems

In design and computation in architecture, fundamental difficulties of developing generative design systems include the multiple objectives typical of an architectural design problem and the considerable size of the search space that contains possible architectural solutions. The aforementioned technical advancements, emerging new building types and uses, and increasing information to be processed, are primary causes of the exponential increase of the search space in architectural design instances. Combinatorics, brute-force search, a process of elimination, and generalization and simplification of problem frameworks are some of the conventional approaches in generative design, but these strategies alone are nevertheless unable to efficiently respond to the aforementioned complex conditions. In the case of adaptable growth models, objectives are also ever-changing dynamic properties of the models, and the computational design systems of such models are expected to possess certain solution-seeking behaviors that can maneuver through the dynamic “solution-scape.”

Development of adaptive design systems may benefit from active implementations of self-organizing logic by gaining its characteristics, such as flexibility, adaptability, and tolerance for growth; otherwise, reconstructions of emerging large-scale structures impose enormous costs and social impacts. In recent years, many scientists have started to obtain the advantages of self-organizing systems in nature through their computational

models (Bonabeau et al., 1999). The main advantages of such systems are robustness, flexibility, adaptability, concurrency, and multiplicity. In the bio-inspired computation field, application of ants' foraging behaviors to telecommunication networks (Schoonderwoerd, 1996; Di Caro, 1997) and control of multiple robots (swarm robotics) (Lipson, 2006; Murata, 2006) are a few examples of applications that use distributed controls to gain more flexibility and robustness in their systems. In this thesis, computational methods inspired by self-organization will be called "self-organizing computation," and detailed descriptions of this phenomenon will be provided in chapter 3.

A question arises from the observation of the recent technical developments inspired by self-organization in natural systems: Is there any place for such applications in architecture? What are the computational methodologies that enable such emergent formation processes? Can such methodologies be applied to development of a computational tool that can contribute to generation of architectural instances? What are the potential implementation areas in architecture?

1.7 Hypothesis

As I mentioned earlier, a bottom-up approach will not be the universal solution for all building types. Many existing building types have very clear scope for future scenarios and no particular future adaptations are required. Existing methods seem to provide efficient controls for a majority of architectural projects. However, once unpredictability and complexity of systems reaches beyond a certain critical limit, top-down deterministic solutions alone may not be able to respond to all the potential future conditions. There are

obvious limitations in existing top-down design strategies and centrally controlled systems when they are called on to support adequate flexibility and extensibility in order to manage the overwhelmingly increasing quantities and complexity of information that are processed during design and construction phases. The hypothesis of this thesis is that the computational implementation of self-organizing principles is one potentially valuable strategy to improve the design of systems that are required to change over time.

In line with Alexander's prediction half a century ago (Alexander, 1964) about the rapidly changing culture of his day, I speculate that there will be more demands for buildings to be able to adapt to newly emerging needs for different qualities and quantities of architectural and urban-scale components. Within the limited allowable growth areas in dense urban settings, the desire to be able to accommodate new needs will increase demand for more adaptable buildings that can avoid future demolitions or drastic reconstructions. As one possible adaptation to likely future conditions, this thesis proposes active incorporation of the self-organizing approach into our existing models of designs.

This approach has considerable potential in architectural and urban-scale applications. I believe that architectural design and construction processes can benefit from the application of decentralized thinking as a core ideological innovation, and this thesis responds to the issues discussed above by proposing self-organizing computation as a strategy to provide possible alternative solutions for architecture and urban design problems.

1.8 Outline of Chapters

Chapter 2 reviews existing examples of self-organization and emergent formations from various natural systems. Then, current states of certain artificial systems, including modular systems and urban-scale city growth simulations, are reviewed and discussed. A critical overview and comparison of natural and artificial systems is made, based on the relative time span and scale of each system.

Chapter 3 introduces the basic framework of computational methods related to the concept of emergence. I categorize computational methods according to three evolutionary stages of applications to design – *evaluation*, *design search*, and *growth and adaptation*. In each category, relevant computational algorithms are introduced and their correlation with existing design precedents is explained. After the definitions of three methods, the concept of *self-organizing computation* is introduced as one strategy of a *growth and adaptation* method. Two examples of self-organizing computation are provided to support the conceptual discussion.

Chapter 4 elaborates on the architectural implementation of two computational methods introduced in the previous chapter: *evaluation* and *design search*. These methods are applied to the framework of architectural design problems with programmatic requirements within spatiotemporal settings. I examine a spring-mass system as an example of evaluation, and a turtle interpretation of L-system as an example of design search. The implementation is unique in its application contexts and is kept at the conceptual level of applications for the sake of logical clarity.

Chapter 5 presents two examples of architectural implementation of the growth and adaptation method. The first shows growth models using diffusion-limited aggregation (DLA); the second shows physical implementation in architecture using a multi-dimensional optimization method. The concept of growth and adaptation is the main focus of the thesis, and the two examples here establish conceptual foundations for a more elaborate application of growth and adaptation in the next chapter.

Chapter 6 integrates all knowledge from the previous chapters and presents a spatiotemporal growth and adaptation system in an urban-scale setting. This system undertakes to simulate evolution of city growth based on the correlation between landform and man-made artifacts. The key concept of the system is bidirectional feedback between environments and subjects under the design. Environments and man-made artifacts such as streets and buildings take coevolutionary paths of development. The results obtained add to our knowledge of how far we can develop design without relying on imposition of knowledge external to a system.

Chapter 7 provides general conclusions and an outline of future work. Advantages and limitations of the methods explored are summarized.

Chapter 2

General Characteristics of Self-organizing Strategies and Their Computational Simulations

Introduction

This chapter reviews relevant examples of emergence and self-organization from natural systems and artificial systems. In addition, I have prepared a separate section to review self-organizing phenomena on the urban scale. Several computational systems that simulate and generate urban growth patterns are also reviewed. After the review, a critical overview and comparison of natural and artificial systems is made based on the relative time scale and the size of each system.

2.1 Natural Systems

2.1.1 Introduction

The existence of a single strong concept or directive as a dominant driving motivation for the building design is often advantageous in architectural design. Working with a strong conceptual direction helps us to conceive a macroscopic framework from the earlier stages of building design processes, and it also helps to maintain cohesion down to the details throughout the processes. This approach often provides consistency in aesthetic qualities. Many signature expressions by master designers have been accomplished by this approach, having clear blueprints and goals from the outset. In general, architectural design processes tend to start by defining macroscopic views by setting clear goals, and then to resolve and elaborate microscopic details based on the initially defined goals.

However, as the number of elements in building programs increases, conforming to an original core design concept while maintaining cohesion throughout all the elements of building design has become a challenge. This approach may itself cause various compromises in performance of buildings, and undesirable allocations of various secondary programming elements can occur inside the buildings. Mere aesthetic preference may not be a good motivation for design of complex structures with greater programmatic requirements. Furthermore, obsession with certain formal aesthetics, compulsive ideological connotations, or preconceptions about spatial relationships can occasionally be dangerous by imposing directives that are external to the essential characteristics of the buildings. To avoid these dangers, either derivation of the final form can be delayed or the fundamentally generative process can be altered at a conceptual

level. The notion of self-organization in biological systems represents completely different pattern formation processes from those of humans. The emergence of many macroscopic patterns found in natural systems occurs through processes and interactions defined at the microscopic level, and distributed behaviors seen in natural systems suggest some potential for new means to adapt to emerging complexity in building industries.

2.1.2 Self-organization

“What is it that governs here? What is it that issues orders, foresees the future, elaborates plans, and preserves equilibrium?” are the famous words of a poet, Maurice Maeterlinck (1927), when he faced the apparent complexities of termites’ organized behaviors. From the standpoints of biology and computer science, *Self-organization in Biological Systems* by Camazine et al. (2002) provides profound insights about differences between design processes in natural systems and those of humans. The definition of self-organization in the context of pattern formation in biological systems is the following:

Self-organization is a process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. Moreover, the rules specifying interactions among the system’s components are executed using only local information, without reference to the global pattern. (2002: 8)

In general, *self-organization* implies a wide range of pattern formation processes in physical and biological systems. Examples listed in Camazine et al. (2002) include sand grains assembling into rippled dunes, chemical reactants forming spiral patterns such as the Belousov-Zhabotinski reaction, cells in slime mold forming highly structured tissues,

lichen growth patterns, pigmentation patterns on shells, fish, and mammals, schools of fish, and a colony of termites building a nest. Roughly speaking, these systems can create patterns through interactions internal to the systems; and most importantly, these pattern formations can occur without intervention by external directing influences. Then, what is the meaning of the term *pattern*? According to the authors, pattern is *a particular, organized arrangement of objects in space and time*. The components or building blocks of these patterns can be living cells or organisms themselves (living units), but they can also be inanimate objects such as bits of dirt and fecal cement that compose the termite mounds. In any case, the common characteristic among the above systems is the ability to successfully build patterns with no external directing influence.

Self-Organization

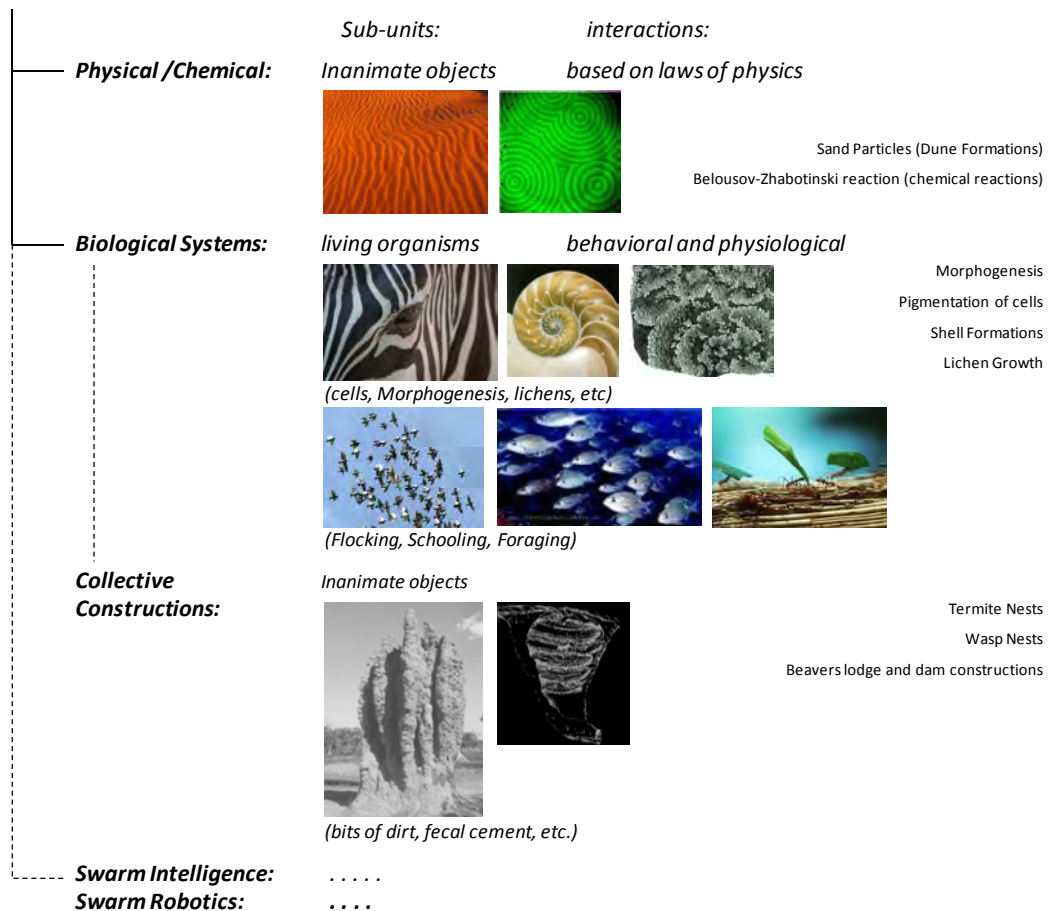


Figure 2.1 – Categories of Self-organizational Systems in Nature

Those patterns resulting from various relatively simpler local interactions can be described as *complex*, but what exactly does the term *complexity* or *complex systems* refer to in precise scientific contexts? The term “complex” is a relative term. The definition provided by Camazine et al. designates “a system of interacting units that presents global properties not present at the lower level” as a complex system. The terms “chaos,” “complexity,” and “dissipative structures” have become buzzwords in articles on non-linear systems. Self-organization in natural systems as introduced in the book can be categorized as one branch of the complexity paradigm in its larger context.

Firstly, Camazine et al. divide self-organization phenomena into *physical systems* and *biological systems*. The main difference between the two kinds of systems is in their interacting subunits. The subunits for the physical systems are *inanimate objects* such as grains of sand or chemical reactants. On the other hand, those of biological systems are *living organisms* such as fish, ants, or neurons; hence they imply greater inherent complexity. Another critical difference is that the patterns in physical systems are induced by interactions based solely on physical laws, whereas those of biological systems can be influenced by physiological and behavioral interactions among the living components. Biological subunits can gain information about the local states of the system and behave according to the genetic programs that motivate the natural selections according to the genetic programs that motivate them. The authors also note that recent findings in studies of self-organization indicate that interactions among system components can be surprisingly simple compared to the highly sophisticated outcomes (patterns) which they can achieve through these interactions.

The self-organizing system – often seen in social insect behaviors – is a *decentralized problem-solving system* that consists of relatively simple interacting entities. In a decentralized system, each individual gathers information on its own and decides for itself what to do based on its local properties, and this activity is carried out in dynamic fashion. Unlike the convergence toward static preconceived goals in some human activities, continual dynamic interactions among simpler lower-level entities produce and maintain the goals. In these systems, the goal itself is an ever-changing property and requires continual interactions. These characteristics make their problem-solving approaches very ‘flexible’ and ‘robust.’ Flexibility in self-organized systems means being adaptable to constantly changing environments. Robustness is their ability to function as a whole regardless of some imperfection in performances at the local lower level of components, which means that failures to perform tasks by some individuals in the system are not always fatal to the entire system’s ability to function. Even though individual entities do not possess sophisticated cognitive capabilities, aggregation of these entities interacting in dynamic fashion based on simple locally distributed rules is able to maintain and direct the system toward the globally optimal solutions. Of course, human systems also possess this type of tolerance for unpredictable conditions to some degree. However, it is likely that the tendency to seek perfection and to gain immediate efficiency based on thorough before-the-fact calculation is more significant in human systems than in biological systems, and there may well be some valuable knowledge to be gained from investigating and adapting those systems’ behaviors. The consequential resulting products of those systems’ collective behaviors are often thought to possess emergent properties.

Emergence refers to “*a process by which a system of interacting subunits acquires qualitatively new properties that cannot be understood as the simple addition of their individual contributions,*” according to Camazine et al. (2002). One dramatic example that displays the intuitive sense of emergence is the phenomenon of Bénard convection cells. Above a certain threshold temperature, initially homogeneously layered cells in laboratory trays suddenly organize into an array of hexagonal cell arrangements, and this change is not a gradual one. A new pattern emerges to form a stable configuration when the amount of heat accumulated inside the cells reaches a certain point. This type of behavior is seen in many non-linear systems, and the emergence of such a pattern or property is called an *attractor*. Much research on self-organization is directed to the use of non-linear systems as models for describing natural systems.

2.1.3 Synthetic versus Natural systems: Pattern formation by humans

In contrast to the aforementioned natural systems, systems that traditionally are executed by human intelligence show entirely different motives and behaviors, and Camazine et al. (2002) display keen observations about the differences between the two systems. The authors point out that there are four ways in which a group of intelligent beings can build an ordered structure without self-organization. The mechanisms which exemplify pattern formation by human groups are *leader*, *blueprint*, *recipe*, and *template*. These four means are externally imposed on the group’s activities in orderly fashion without the interactions of the system’s components seen in self-organization. The existence of a well-informed leader is quite a common characteristic in human group activities, as majorities of our corporations in societies possess top-down hierarchical organizational

structures. Many constructive processes essential to our activities are more or less multiple combinations of the above listed mechanisms, such as a leader and a blueprint. A recipe provides the step-by-step instructions for the pattern, while a blueprint provides only what is to be built. A recipe can be used with behavioral adjustments based on the feedback from the emerging pattern, and some of the human activities represent this capacity to flexibly adopt changes to some degree in combination with relatively more centralized strategies. An example of a recipe can be literally a chef tasting and adjusting the seasoning of his dishes as he cooks. A template is a full-scale guide or mold that specifies the final pattern, and the authors list cookie cutters and candle molds as typical examples.

Camazine et al. (2002) also speculate about why many animal species do not rely on blueprints. For example, mental blueprints for complex structures, such as termites' nests, consists of a vast quantity of information, and genetically encoding the information and storing holistic pictures inside each of an individual's genes is extremely "expensive". Reliance on blueprints for animals entails each group member to be able to interpret and extract the building operations out from them, and this is beyond many animal groups' individual mental capacities. This proneness to more distributed task management among animals becomes clearer when the aiming pattern gets larger and more complex. For instance, the pattern formation process requiring sequential stages of operations is very difficult for animals to conceive solely from the blueprints. Perhaps the highly evolved human cerebral cortex allows us to develop dynamic interpretations by high level of mental sophistication, and each individual can figure out instructions from the blueprints. All the above facts are provided from disciplines outside of architecture or design. It is a

little surprising to find that such philosophical observations about essential human construction activities are rarely discussed among designers currently.

2.1.4 Collective Construction



Figures 2.2 – Collective Constructions by Social Insects (Termites and Wasps)
(From http://en.wikipedia.org/wiki/Image:Termite_Cathedral_DSC03570.jpg)

In contrast to construction processes by humans, the *Collective Constructions* seen in nature, accomplished by wasps or termites, indicate the existence of fundamentally different construction principles based on completely different logics and behaviors. Firstly, wasps or termites do not have awareness of the global goal of their constructions. Unlike human constructions, no predetermined blueprints are available throughout the constructions, and it is doubtful that there is ever any awareness of the final configuration or convergence in their minds during the construction. Their process does not depend on supervisors or central leaders monitoring their progress and giving instructions. The critical difference from the conventional centralized human design method is that the

self-organized internal logic inside the system is able to search and design without knowing its transcendent global target in a bottom-up manner, rather than casting and limiting the final form or gesture in a top-down manner before fulfilling all the requirements.

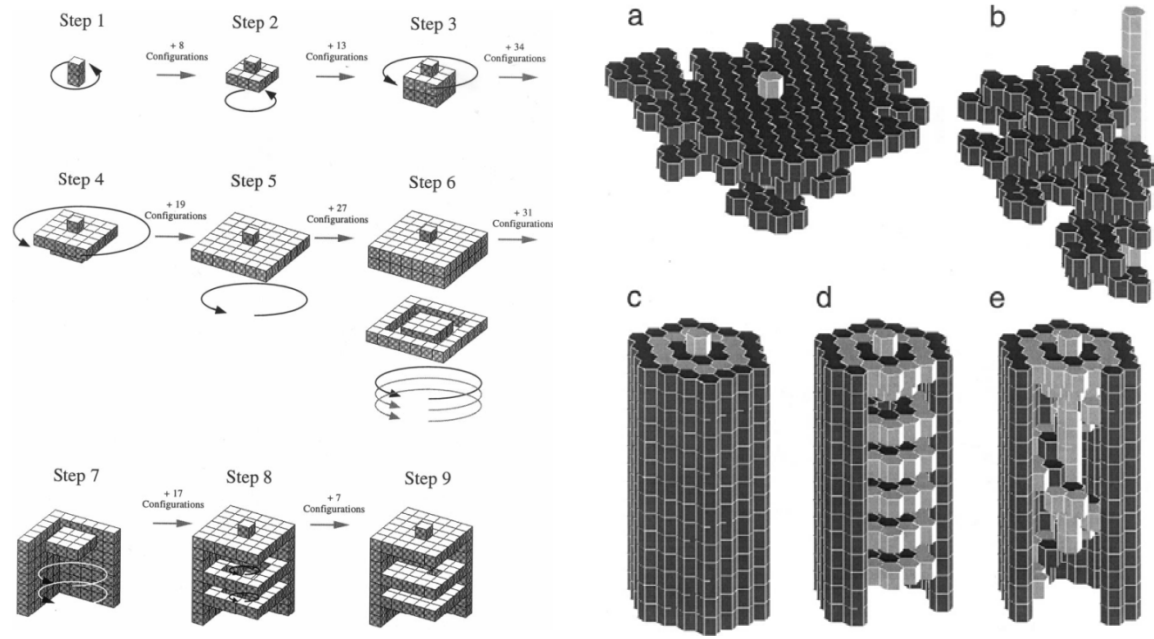
These processes are completely different from many existing conventional design processes which we have seen in human architecture so far. Our design process does not flexibly change its goals during its process, and oftentimes blueprints or global gestures are known in advance or imposed by leaders at the beginning of their design stages. Although we have a collaborative process among architects, engineers, and various consultants, a goal and fundamental objectives for design rarely exhibit dynamic changes during the course of design. The existence of a well-informed leader is a quite common characteristic in human group activities, and a majority of our corporations in society possess top-down hierarchical organizational structures. In contrast, design processes in wasp and termite colonies are self-organizing processes, and their final formal configurations are the results of optimization by locally distributed multiple intelligences.

One well-known example of these self-organized, distributed behaviors is the beaver's ability to build lodges and dams from branches, mud, and other debris. According to Camazine et al. (2002), beavers do not seem to rely on any innate concept or blueprint of the structures they build. Instead, the authors speculate that their building behaviors are genetically programmed responsive acts which are triggered by the beavers' surroundings. This kind of stimulus-response is often called *Stigmagy* (Grasse, 1959) by researchers. *Stigmagy* is an important notion for understanding the process of collective constructions.

In this notion, information from the local environment under dynamic progressions stimulates and guides further activities in construction. A certain local state of the system becomes an incentive for the next construction for individual workers, and this process continues to feed new information to the builders. In this way, information is always provided from the dynamically changing environment rather than any source of information external to the ongoing construction activities. This is one of the reasons why social insects, such as termites, can undertake complex constructions without knowledge of the ultimate form of the structures. Thus Stigmagy often refers to the information collected from works in progress. Consequently, the resulting products of these collective activities are often thought to possess emergent properties.

Computational Interpretations

Theraulaz and Bonabeau are two of the pioneers who have provided the computational interpretations for the logics behind the collective constructions by wasps (Theraulaz and Bonabeau, 1995a, 1995b). Their generative methods are based on three-dimensional cellular automata with locally defined rules, and agents move randomly inside the lattice to drop the building blocks based on the cellular neighborhood rules. This interpretation of wasp behaviors is based on the notion of precedent Stigmagy (Grasse, 1959). In this purely conceptual experimentation, the ultimate objective is to gain cohesive structure solely from locally assigned series of construction rules without providing any global information about the structure (such as blueprints). The wasps in their model do not have any global sense of what they are constructing; instead, they have their local sensing as a stimulus to continue their constructions.



Figures 2.3 – From “Modeling the Collective Building of Complex Architectures in Social Insects with Lattice Swarms” by Guy Theraulaz and Eric Bonabeau (1995a).

The experiment takes place in the hypothetical lattice space inside a digital environment, and numbers of construction agents are distributed inside the lattice space. Series of rules (instructions) indicating the proper placements of construction blocks, based on the 3 x 3 x 3 neighborhood conditions, are originally randomly generated and given to the agents, and they place the building blocks as they find the stimulating configurations which match the given rules. With this method, they can continue to construct the structure based solely on the feedbacks from local dynamic neighborhood conditions in concurrent fashion. By iterating this process for a certain period of time, agents will produce structures corresponding to the given rules. By sequentially applying different sets of rules step by step, they successfully direct their agents (wasps) to construct even more complex structures which resemble the actual nests constructed by *Epipona* wasps.

Here, the experiment is taking advantage of spatiality and the emergent property of cellular automata, and the stochastic nature of Monte Carlo methods. The randomly walking agents are used to add some noise and variations for the resulting architecture. Perturbations in results also depend on the types of rules wasps apply. Based on the rules and the combinations, their resulting structures can be either deterministic or non-deterministic and occasionally have some probabilistic variations in their growths, since the movements of agents are not defined deterministically. Some sets of rules are deterministically applied regardless of the agents' random movements, and they always create the identical resulting configurations; while some sets of rules allow several different resulting configurations based on which areas in the lattice are stimulated earlier during the process. Potentially, design tendencies or the likeliness of certain patterns are describable with a few rules instead of requiring complete resulting forms. Use of a relatively abstract form of design description, 'Rules,' is suggested in this method.

Auto-generation of building structures

The next step for the experiment is to auto-generate and select the fittest set of rules by providing evaluation criteria for arbitrarily produced populations of building structures. Experiments by Theraulaz and Bonabeau's more recent paper (Bonabeau, 2000), and also the present author's personal design exploration, represent some efforts to computationally carry out the abovementioned step.

From the author's experiment, construction by computational agents using every set of rules is performed until the structures cease to grow or the numbers of iterations exceed a certain threshold. Populations of sets of rules are compared and evaluated according to

the resulting structures. Then, based on evaluation, rule-sets that score higher fitness values will be considered as elite sets and remain as parent sets for the next generation's schemes. For every rule in every set, the numbers of times that they are stimulated is recorded, and the rules that are used more often are ranked higher among the rule-sets. The rules that are never stimulated during the simulation will be discarded from the rule-set (extinction). New generations of rule sets are produced by simple cross-over between aforementioned parent sets, and a mutation process adds a randomly generated new matrix of rules to prevent the search from stagnating within local search spaces. The above iterative process is based on the genetic algorithm (GA) developed by John Holland (1992). For the sake of clarity, the fitness criterion that was chosen for the experiment was based on simple local checkability. The aim is to direct the evolution toward the formation of larger clustering patterns. For every block in the resulting structures, the numbers of identical matches of local neighborhood cellular configurations are checked, and the matches in the neighborhoods with the larger radii (up to 5 units) are considered to have higher fitness, as it implies similarities on a larger cluster scale.

From the author's personal experiment, the results indicate the appearance of some coherent structures in an earlier generation (Figures 2.4, 2.5, & 2.6). We recognized self-similarities in some structures which resemble the Sierpinski triangle. Once the numbers of blocks exceed certain numbers, growth seems to converge on simple filling of the lattice or alternating grids. Dividing the above fitness results by the number of cells generated is one strategy to penalize excessive growth, and to avoid high-density solutions. Using hard-coded rules that guarantee the generation of coherent structures as the GA's initial population, instead of randomly generated rules, is another strategy.

However, initial rules are gradually replaced by cross-over and mutation after a few generations and eliminate the initial population's influence on geometries. Due to the purely computational nature of the experiment, selection of proper fitness criteria has become a profoundly complex issue beyond its architectural interpretations. What appears to be coherent in numerical format does not always give us visually recognizable patterns. This discrepancy between our intuitive visual perception and our numerical implementations requires further study and will be an interesting challenge for future explorations. This fitness criterion can, theoretically, be designed carefully to implement complex conditions to assimilate real-life scenarios in architecture.

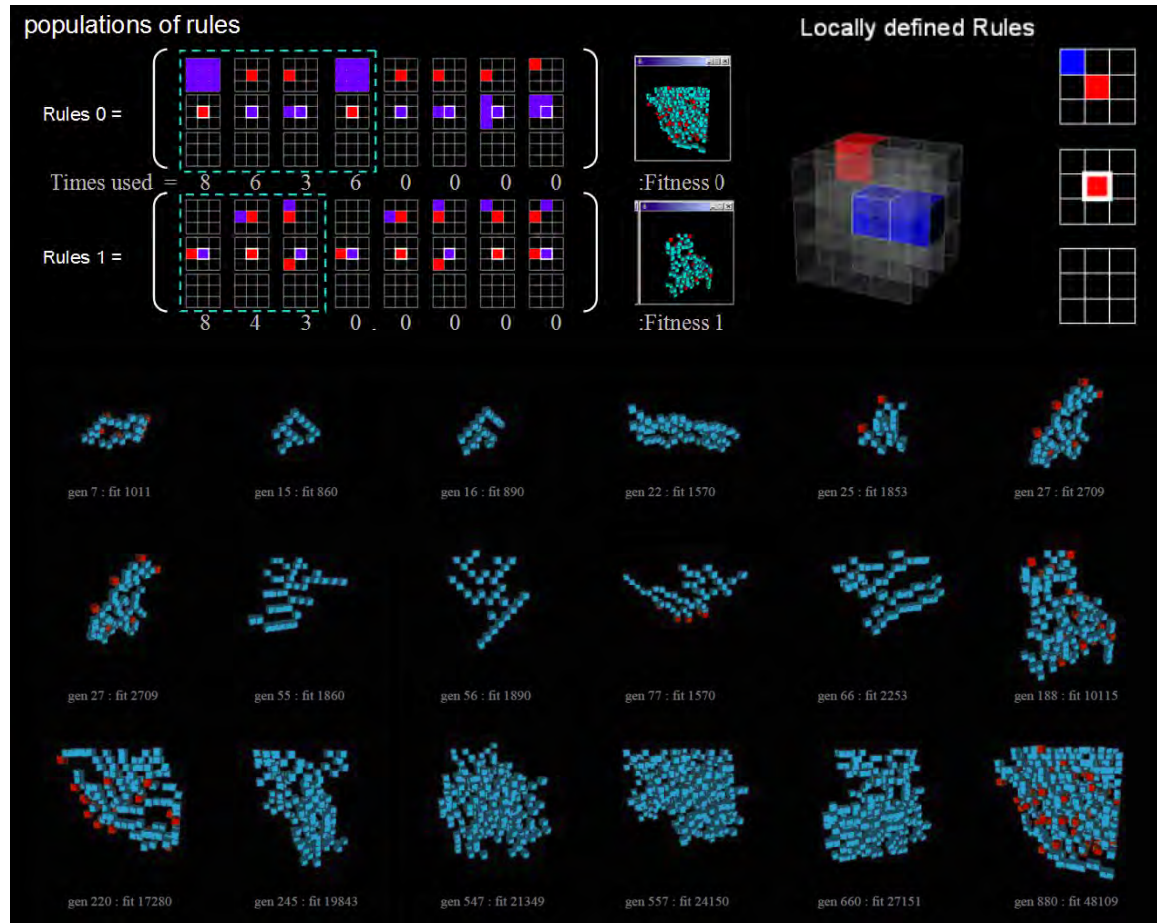


Figure 2.4 – Sets of local rules to grow structures (top left). Procedural Representation of design and its evolutions (top right). Examples of Structures evolved autonomously in evolutionary run. Configurations are based on the fitness evaluation (Bottom). (Figures produced by the author.) (Narahara, 2008)

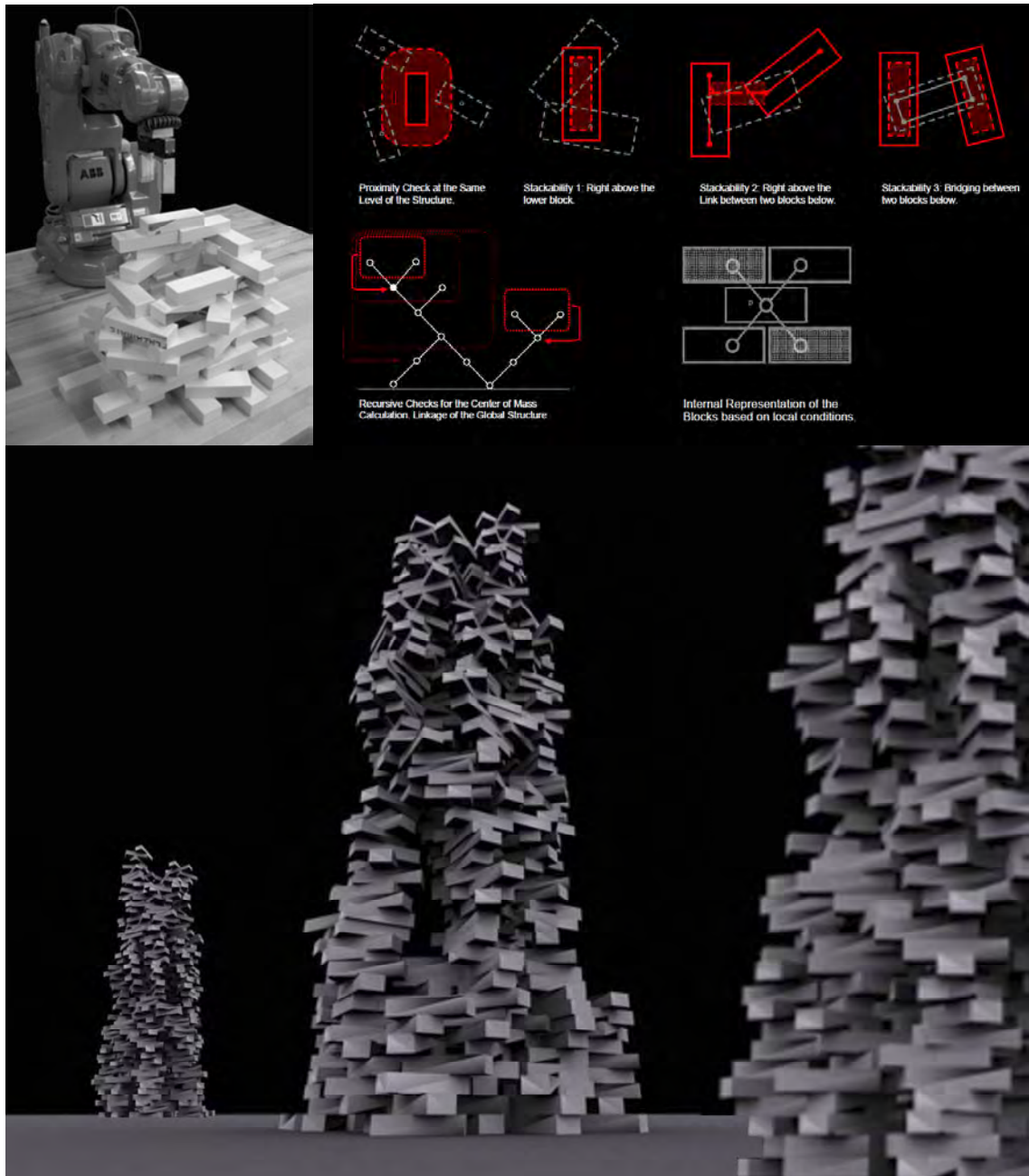


Figure 2.5 – Generative Grammar Sets developed based on the Constraints of the robot’s (RB-140 by ABB co. Ltd.) gripper movements and the Stackabilities of the blocks. Experiment was inspired by the experiments from previous page (Top). (Figures produced by the author.) (Narahara, 2008)

Figure 2.6 – Possible generative construction iterations using unit blocks based on the hypothetical fabrication constraints of the robot (Bottom). (Narahara, 2008)

2.1.5 Applications of Self-organization: Computation Methods related to Emergence

In the area of computer science, a large number of scientists have started to pay attention to self-organizing systems, particularly to those seen in the social insect behaviors, and they try to take advantage of these systems when solving our problems. This approach focuses on the characteristics of self-organization, such as distributedness, flexibility, robustness, and interactions among relatively simple agents. A number of applications have been in the areas of combinatorial optimization, communications networks, and robotics, all inspired by self-organization. The social insect metaphor can be well linked with the formulation of artificial intelligence and swarm intelligence. According to Bonabeau et al. (1999), *swarm intelligence* refers to *the emergent collective intelligence of groups of simple agents*. Bonabeau stated that this application is important because it will offer alternative ways to design intelligent systems driven by emergence and distributed functioning to replace conventional centralized control and programming. This trend and the interests in self-organizing systems among computer scientists motivate them to model and simulate social insects' behaviors. Eventually, their goal is to design artificial distributed problem-solving devices that self-organize to solve problems based on hints and inspirations from the social insect behaviors. The current popularity of object-oriented structures among computer programs, or the rapid growth of web-based information systems, such as Wikipedia, can be an indication of our growing interest in distributed ways of thinking across disciplines and beyond specific academic communities.

One of the first approaches to mathematically and computationally understand emergent behaviors was accomplished by Turing in the 1950s. He wrote the seminal paper in the area between biology and computer science, called “The chemical basis for morphogenesis” (Turing, 1952). In this paper, Turing was trying to provide a mathematical interpretation of morphogenesis. The model that Turing proposed was one of the first to model an emergent pattern formation from interactions and diffusions among more than one (multiple) chemicals, and this notion is considered to be one of the earliest examples of biological models described as complex systems. He designed the systems with two chemicals – activator and inhibitor – in a correlating relationship, and the faster diffusion rate of the inhibitor induces the formation of patterns which are often called Turing patterns. It is known that a variety of patterns can be generated by changing parameters in the Turing equation, and this logic of pattern formation gave a basis to explain patterns in nature, such as pigmentation patterns on shells, fish, and mammals. This direction established by Turing and some other scientists has been further developed by many researchers such as Prigogine. They performed studies of dissipative structures under the influence of non-linear interactions (Prigogine and Stengers, 1984).

The Turing’s mathematical interpretation of morphogenesis was based on differential equations and meant to provide analytical interpretation of the phenomenon. However, there are several different up-and-coming techniques and methods for formulating models of emergent systems to simulate various collective activities. For example, Camazine et al. (2002) listed three common approaches: differential equations, Monte Carlo simulation, and a cellular automaton (CA). The same collective activities can be modeled by different methods, though they each have their own pros and cons. The differential

equation models are more explicitly described with mathematical formulas (such as Turing's equations) and represent each entity's (individual's) behaviors over time by precise quantitative and rather deterministic descriptions. Many scientists prefer the analytical clarity inherent in mathematical expressions.

On the other hand, a Monte Carlo simulation can treat an individual's behavior probabilistically, and there is more room to implement perturbations or noise among the interacting entities. This can be an advantage when systems display more whimsical behaviors, such as insect moves. This method often requires extensive calculations to execute a large number of iterations to gain viable results due to the probabilistic nature of the simulation. Unlike differential equation models which can deterministically derive results, the Monte Carlo simulation of ant foraging needs many trials to determine the equilibrium distribution of the ants. Camazine et al. (2002) indicate that the average results of all runs from the Monte Carlo simulation for their ant foraging model eventually converge on the results from that of differential equations as they increase the number of iterations.

A cellular automaton (CA) can describe 'spatial relationships' by arranging subunits over a discrete grid or lattice. Interactions between the subunits are described by simple cellular neighborhood rules, and based on these rules various subunits will change their states over discrete steps of time. The above two methods are fundamentally different from the models using differential equations which describe continuous process mathematically and provide analytically convincing results.

Many models of emergent systems are not always analytically solvable or describable, and it is occasionally necessary to rely on numerical calculations or optimization techniques by computers. This is where techniques such as Monte Carlo and CA become potentially valid and useful. Due to a recent dramatic increase in desktop computational power and speed, Monte Carlo and CA simulations have become more promising techniques for researchers, as both require computing a large number of iterations to determine the results.

These two contrasting and distinctively different attitudes towards problem solving exist in the understanding of many phenomena in our daily life. In almost any field of studies, not only engineering but also social science, economics, and architecture, this duality of problem-solving attitudes exists. The analytical approach is much more deterministic in terms of its methods and results, whereas approaches that rely on computational iterations using stochastic simulations, such as the Monte Carlo methods, are the “trial and error” type approach of a heuristic process that relies on random samplings. Analytical and deterministic problem-solving methods are not always available for all the problems. When one cannot formulate explicit equations for problem-solving methods, the latter approach based on heuristic processes can be a good alternative approach.

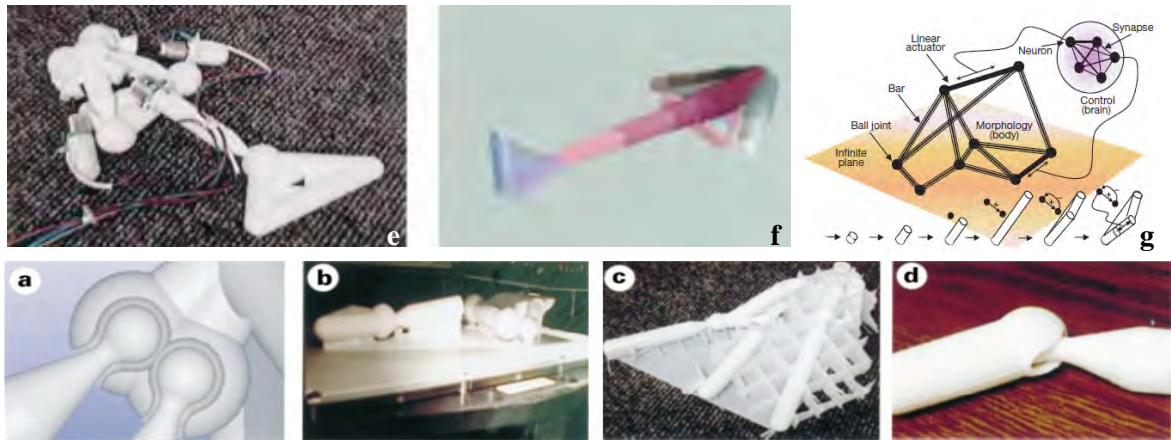
The best known example indicating this duality of problem-solving approaches is Buffon’s needle experiment in 1777, an example often used to introduce the Monte Carlo methods. Basically, the probability of a needle to fall between the two color stripes of a surface (or onto one color) can be derived quite deterministically by using geometrical probabilities. In this way, without committing any trials, one can analytically derive the

regulating threshold value. However, another approach to solve the same problem is to simply throw as many as needles as possible and gain the results empirically, based on this stochastic approach. Similarly, queuing models for various situations such as ticketing counters at airports can be modeled both analytically, using Poisson's distribution for the arrival rate of the incoming people, and stochastically, using Monte Carlo simulations. The results from Monte Carlo simulations are known to converge toward the results from the analysis. Moreover, Monte Carlo simulations are used when no kind of deterministic algorithms is feasible, or is incapable of modeling phenomena with many uncertainties in inputs. Obviously, the aforementioned duality does not exist in all problems – especially for the modeling of emergent phenomena.

A genetic algorithm (GA) is a heuristic optimization technique that uses randomness and is a computational methodology related to the concept of emergence. A GA is categorized as an evolutionally computational method, and possesses the interesting characteristic that globally functional solutions can be evolved by resolving local neighboring conditions in a repetitive manner. Similar to the Monte Carlo methods that use the random samplings, a GA uses stochastic selection in order to form a new population. (Detailed description of a process using a GA will be provided in the next chapter.) Besides being reliable search optimization strategies for problems such as the Traveling Salesman's Problem (TSP), a genetic algorithm has been used for generative purposes, for example, for artificially evolving physical electrical circuits, for finding better configurations for antenna designs, and so on. The critical difference from conventional deterministic and analytical problem-solving methods is that a GA does not require instructions about how to find the solutions. By applying random genetic

operators, it can perform variation and selection, and it is possible to determine unknown solutions to complex problems to some extent. This is a common characteristic among evolutionary algorithms which rely on the amplification of fluctuations (randomness), and this characteristic enables the discovery of new solutions. In addition, a GA does not make any assumption about its solution space (fitness landscape) at the outset of its search process, and this generality of a GA allows it to be used in diverse fields beyond science and engineering. In architecture, typically solutions for multiple requirements from programs are unknown, and there are no deterministic search methods. Therefore, proper applications of a GA are a potentially a good approach for approximating solutions for many architectural problems.

This unique characteristic has been utilized by researchers not only for the purpose of problem solving but also for searches toward the unknown or undefined goals. Some of the computational experimentation by leading scholars using GA and GP (genetic programming) represents attempts to synthetically assimilate the evolution of artificial life forms within a computer environment. For example, H. Lipson (2005) used a genetic algorithm to evolve configurations for robots with basic building blocks such as bars, actuators, and artificial neurons, to find the fittest machines based on the locomotive abilities of the robots inside the virtual environment. Another current trend using evolutionary computation techniques is a design of adaptable systems that can dynamically reprogram themselves to tolerate various unpredictable changes of environments. These highly speculative experiments are beyond the domain of simple simulations of existing physical phenomena, and display an ambitious approach to evolve and foresee the solutions beyond discernible domains of our search spaces.



Figures 2.7a-g – Diagrams from “Automatic Design and Manufacture of Artificial Lifeforms” (Lipson, 2005).

2.2 Artificial Systems

2.2.1 Reframing Human Design Activities

This thesis is about the investigation (and the exploration) of fundamental components in architecture: physical or non-physical components that form and organize spatial order in response to emerging sociological needs and technological innovations.

The search for indivisible components that make up the entirety of the universe of matter has been a perennial agenda for human civilization. Even in the pre-Socratic era, the concept of ‘atom’ existed. Leucippus and Democritus were the first ones to propound the theory of Atomism, that all of reality is made of indivisible building blocks. Among contemporary physicists, the question, “Is matter infinitely divisible?” is still an unsolved mystery. This question can be paradoxical, since the absence of observation of such an element could never prove its absence; thus we may never understand the truth.

The concept of finite composing units originated by early philosophers such as Democritus has some importance for human design activities. Almost all man-made products can be seen as a composition of discrete finite elements. The agglomeration of primitive components constitutes a unique composition, and how we organize components will make differences in resulting compositions. A brick in masonry construction is a good trivial example of a basic building unit in architecture. The search for appropriate basic building blocks could help our understanding of creative processes in architecture.

However, in the context of architecture, the search for the indestructible building block, ‘atom,’ might not have direct importance for designers’ production processes. Nobody starts designing a city by selecting hardware for doors. Instead, we usually find better ways to represent each individual building as a block or mass. Selection of these finite units is all relative to what we design, and we usually unconsciously encapsulate large amount of information (data) into a ‘type’ or ‘object’ in order to handle these objects as if they are single components. Abstraction of components helps us to *focus on designing relationships* among various units. This concept of encapsulation or modularization is fundamental for many existing design assistance tools from ancient architectural drawing techniques to contemporary CAD systems.

One example of this can be seen in recent Building Information Modeling (BIM) tools such as Revit (AutoDesk, 2010) which has extensive predefined libraries of frequently used architectural components such as walls, windows, and doors. These components are parameterized, and they can resize and add necessary details in a semi-automated manner

to raise architects' productivity. In addition, some manufacturers provide downloadable components of their products from their website. Architects can focus primarily on designing the relationship among these abstract components rather than spending time detailing every single component. This so-called "object orientation" has been an important concept of recent design culture.

In construction technology, the development of components such as modular walls and partitions is a good example of discrete elements intended to economize on our installation, assembly, and overall production processes. This notion of the 'unit' is the basis for modular systems and prefabricated constructions, and units even became a creative driver for development of architectural styles such as Metabolism, proposing replaceable modular building capsules.

This phenomenon of encapsulation is observed in many contemporary practices across disciplines. Object-Oriented-Design is one such strategy that has been famously applied to modern programming languages. It is a well-organized platform to build a complex network of data structures. This is especially true if multiple individuals are involved in production processes. One reason for this is its ability to encapsulate a large amount of data into a single '*object*.' Once objects are defined, they can be instantiated without requiring users to know every detail of their internal descriptions. Users can define each object and interactions among them, and they can build relationships among objects without going into meticulous details of each object.

In architecture, this type of language setting can potentially produce more flexibility and interactivity among designers and users throughout the design process. The appropriate definition of basic building blocks can help architects to develop their designs more efficiently, and this concept of ‘encapsulation’ can be applied to general planning of architecture as well as to the physical design of construction components. This concept of object-orientation can potentially enhance extensibility of systems in architecture.

“A part and a whole” is an important point of view for designers, especially for architects who are in need of understanding relationships among various contextual elements at far larger scales than product designers. One’s perception of a whole can be a part of a larger whole, and the organizational principles behind a complex web of relationships among buildings are not always obvious. Better understanding of dynamical relationships of ‘global to local’ and ‘local to global’ has been gaining unprecedented importance among us today.

Our processes of building design have a tendency to proceed from macroscopic (global) views to microscopic (local) views. Normally, we organize overall relationships among different buildings on site and fulfill various required programmatic elements inside the buildings in schematic design phase. Then in design development and construction document phases, we gradually resolve more detailed issues. On the other hand, many systems directed by self-organization, which we have reviewed in recent sections, have a tendency to acquire global patterns from the interactions among the system’s constituent units based on purely local information. Empirically, we already have basic organizational typologies for conventional building types, and even some unconventional

building types can be resolved by hybridizations of knowledge we already have. However, increasing complexities and quantities of building programs from newly emerging usages and social demands and new technologies that allow unprecedented spatial organizations have started to demand that architects find new hierarchies and organizational principles which cannot be conceived simply by manipulating knowledge from traditional building typologies. An approach based on ‘local to global’ thinking is an alternative strategy to find unknown organizational solutions.

One of the primary interests of this thesis, ‘adaptabilities in architecture,’ can be achieved by clever use of organizational schemes in order to improve our dynamics of building activities. Participatory designs led by some architects, such as Lucian Kroll and Christopher Alexander, showed examples of such design strategies by providing multi-participatory platforms for designers and clients. Their processes were based on manual and analogue procedures without using computers; however, their concepts, such as Alexander’s pattern language, left some influence on later conceptual developments of object-oriented programming languages. Whether their results are successful or not, their design processes based on group dynamics and multiple user interactions led them to acquire a different design methodology and outcomes from mainstream modernists’ architectural approaches at the time. I will explain more details about their practices in a later section. Their approach is in contrast with the approach of the Metabolists, who had tried to achieve similar objectives by physically replaceable/reconfigurable systems. I argue that the development of physically reconfigurable components can potentially contribute to and reinforce design strategies based on group dynamics by allowing

building components to concurrently adapt to newly emerging needs even after the completion of the building structures.

One provocative example which enabled itself to acquire a living organism-like adaptability was Kowloon Walled City in Hong Kong. Kowloon Walled City displayed incredible flexibility and capacities to transform itself, within a relatively short period of forty years, from a place with a few thousand inhabitants to a structure in which more than 35,000 inhabitants resided. This is one of the closest approximations to a man-made structure that uses self-organization principles such as those found in natural systems. In fact, we are only at the beginning of the process of finding technological and ideological improvements to synthetically achieve such a system. None of the examples by professionals, including Metabolists and practitioners of participatory design, who had envisioned notions of architecture as a reproductive system, have ever reached the emergent quality of Kowloon. In this case, the resulting structures were not necessarily the best examples of architecture for human living, but the structures' ability to reconfigure themselves to accommodate radically increasing their population within a relatively short time was extremely unusual. Incredible plasticity displayed by the structures might have been contributed to by use of relatively low-quality construction methods and materials. Impreciseness and rawness of composing elemental units probably also allowed local residents to have design and construction occur spontaneously in a relatively short time and enabled their structures to gain a fluid-like, smooth transition between one state of the structure to another, to some extent. (More detailed explanations will be in a later section in this chapter.) However, to think of low quality and lack of sophistication as necessary conditions to materialize emergent self-

organizing structure may be misleading. Kowloon Walled City had many practical problems concerning lighting and circulation. But one primary interest of this thesis is whether an active adaptation of a computational approach can potentially improve and enhance the emergent characteristics found in Kowloon Walled City to a practical level of application.

Recent advances in technology have been offering a good opportunity for us to reexamine what appropriate basic building units can be in both design and construction processes in architecture. Reconfigurable modular systems proposed by various computer scientists, such as Lipson and Murata, show the potential for building units to be actively responsive components rather than inert static objects (Lipson et al., 2005; Murata, 2006). They also use decentralized controls based on local communication between modular units, and this characteristic is providing robust and flexible organizational strategies and design processes for global configurations. My hypothesis is that eventually sufficient computational power, material and structural innovations, and advances in communication devices will lead us to conceive extreme forms of adaptability in either synthetic or organic matter. With that in mind, what we can optimally gain in order to take full advantage of decentralized notions within the reach of current technology has become an agenda for the thesis.

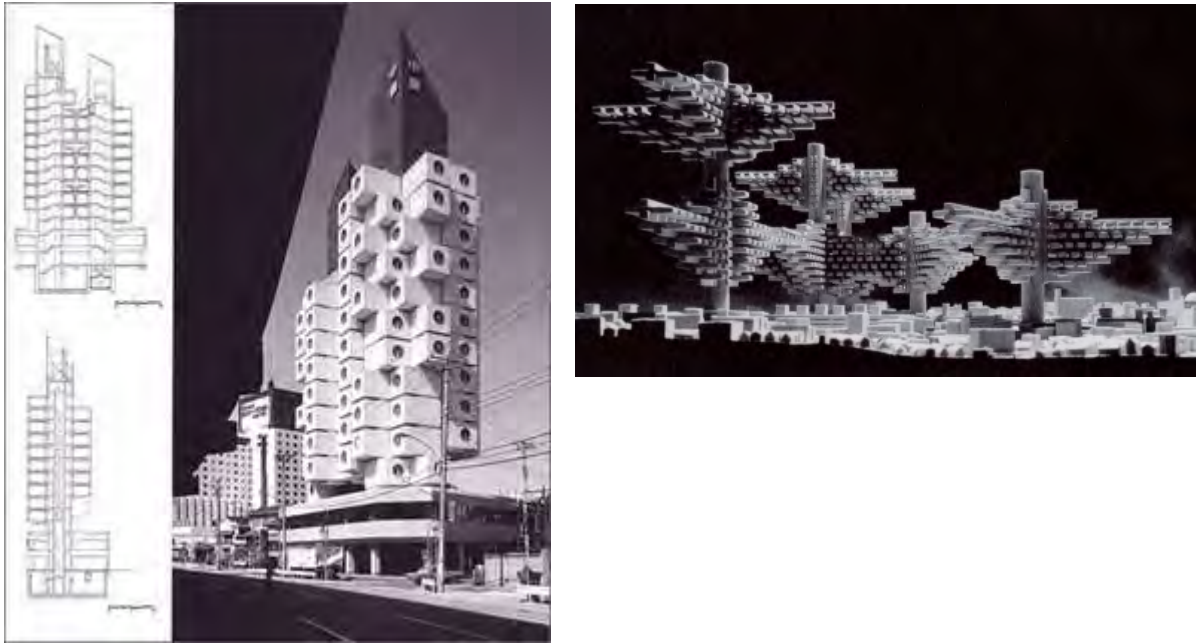
2.2.2 Physical Implementations: Mid-to-large building scale applications

Human efforts to practice Self-organizing Design

In architecture, there have been efforts to create systems that can tolerate programmatic and environmental changes over time. Some responded to these issues by proposing physical systems that can alter their morphological states in order to adapt. In addition to practices in architecture, there are speculative experiments by computer scientists and engineers inspired by self-organizing adaptable growth models seen in natural systems. In the following section, I would like to review those endeavors.

2.2.3 The Metabolist movement during the 1960s in Japan

One of a few examples of architecture representing the principles of emergence, self-organization, growth, and adaptation is the Japanese Metabolists movement in the 1960s, led by rising architects of the time such as Kisho Kurokawa and Kikutake Kiyonori. (Yatsuka, 1997) In the late 1950s, Japan started to experience the pressures of overcrowding in urban areas, and in response to this social issue, the Metabolists proposed plug-in megastructures that could constantly grow and adapt by clipping prefabricated pods onto infrastructural cores of skyscrapers as if they were living cells. To date, very few Metabolist concepts have been realized, yet some buildings – including Kurokawa's Nakagin bachelor capsule tower, built in Ginza, Tokyo, in 1971 – have revealed their unique visions, as well as limitations, quite clearly (Figure 2.8). Original visions of metabolic growth and adaptation were rarely realized physically, as the size and weights of the pods were practically very difficult to reconfigure.



Figures 2.8 – Nakagin capsule tower by Kurokawa, Tokyo, 1971 (Left). (Yatsuka, 1997). The cluster in the air by Isozaki, 1962 (Right). One of the seminal works by the Metabolists.

Scale and size of subunits

One of the fundamental problems that the Metabolists had faced during the 1960s was the mobility of the primary composing elements (units or capsules). These subunits are normally attached to an infrastructural core (tower), and their joints to the core (connection) allow them to have several different options for their orientations. However, these systems do not promise greater numbers of configuration patterns for global structural forms, as the unit's joint has only a few options: to be either detached, joined, or joined with a different orientation. The Nakagin Tower by Kurokawa was a good example of this characteristic. Subunits in Metabolists' buildings are large and static so that use of heavy industrial cranes is required to move and reconfigure the units. Eventually all the systems of the units, such as data connections, plumbing systems, and heating and cooling systems, have become obsolete, and these buildings were never used in the way that they had initially been intended to be. Eventually, some Metabolist

buildings were demolished like any other conventional buildings in spite of their original concept of metabolic growth. Development of flexible and adaptable architecture has been a perennial theme among practitioners, and some of the failures among the Metabolists in the 1960s clearly indicate the difficulties of designing universal subunits that can endlessly tolerate the technological, environmental, and circumstantial changes associated with structures.

2.2.4 Isozaki's Concept of Process Planning

Arata Isozaki was not officially a Metabolist member (Yatsuka, 1997). However, he offered an interesting manifesto about time-based architectural design methodologies in “Process planning” (Isozaki, 1967, 1972). In it, he listed two categories of architectural design planning methods: closed planning and open planning. Closed planning is a relatively traditional planning style that anticipates a complete final product of design from the outset of a production process. In this case, the end of the production phase is the completion of the product, and it has no plan for further extension. The production is based on a precisely planned blueprint which includes all possible requirements up to a certain future stage of time. In contrast, open planning aims at designing a system that can be upgraded for various future extensions from the beginning of the planning. According to Isozaki, this is a characteristic seen in many modular systems in architecture. In this planning approach, most of the efforts are spent on the design of generic or basic components that can accommodate as many conditions as possible. As a consequence, composed spaces become homogenized and lack dynamic extensions. Isozaki speculated that in the near future architectural planning would require placing

more priority on time-based dynamic adaptation for ever-changing environments, and he named this third category as “process-planning.” Many buildings by Metabolists fall into an open planning category. The Oita prefectural library project (1966) by Isozaki, which was claimed to be designed by a process planning concept according to Isozaki (1967), is composed of a systematic tubular structure with well integrated mechanical systems. This appeared to be quite similar to the conventional modular system both in architectural style and scale, and it is similar to those of Metabolists. Perhaps, the true realization of the process-planning method has not been realized yet, even for Isozaki.

Closed and open planning is also a common concept in software engineering. Encapsulation of certain code sequences allows us to use them as a library, and guarantees certain levels of extensibility for software development. As I mentioned in an earlier section, object orientation is a good strategy for building open planning frameworks for systems. Modern concepts of software engineering are shifting towards open planning, and more away from closed planning, as they are encountering constant needs for revisions and updates of their contents and functionalities.

2.2.5 Construction Automation during the 1980s in Japan

Applications of robotics technologies to architecture are relatively new attempts among a few people in this field. Due to their high initial investment cost and the nature of their experimental characteristics, robotics applications in architecture have been considerably influenced by the economy of the time. One of the notable practitioners of robotics technologies in architecture is Santiago Calatrava, and some of his works have kinetic

components that are actuated by robotic control systems. While types of works represented by Calatrava shows applications in building operations and even aesthetical effects associated with the kinematic components, applications found among Japanese general construction companies in the 1980s have focused on efficiency and economy of construction sequences through robotics technologies. Many research studies in this area have been focusing on developing construction automation systems which can achieve efficiency in production processes to compensate for the predicted future decrease of manpower in the construction industry. During the 1990s in Japan, with the help of an economic boom now, in hindsight, called “the bubble,” major Japanese general construction companies competed to develop ambitious large-scale construction automation projects such as ‘Big Canopy’ by Obayashi Corporation (Shiokawa et al., 2000) and ‘Smart Systems’ by Shimizu Corporation.

Those systems are for the construction of a complete building using automated construction and robotic construction technologies. Obayashi Corporation’s Automated Building Construction System (ABCS) in 1993 was one of the earliest working systems in the industry. In the ABCS system, a temporary housing that contains all the production equipments, including robotic cranes and welding robots, is attached at the top of a building under construction, and the entire system is lifted up as they finish constructing the lower floors. The system aimed at reducing construction time and labor needs, and improving quality and safety. T. Shiokawa (2004) indicated that ABCS achieved about a 60% man-hour rate reduction compared to that of construction of a building with the same scale executed by conventional construction methods and schedules. (The man-hour rate is the number of workers times working hours per day divided by square meters of

gross building area. The man-hour rate shrank from 7.8 to 3.2 through switching from conventional methods to ABCS (Figure 2.9).

Although these ambitious projects in the 1990s faded away due to the economic upheaval in Japan at the time – ‘the bubble crisis’ – the essential notions of automated repair and fabrication can still be viable concepts for the future of architectural design.

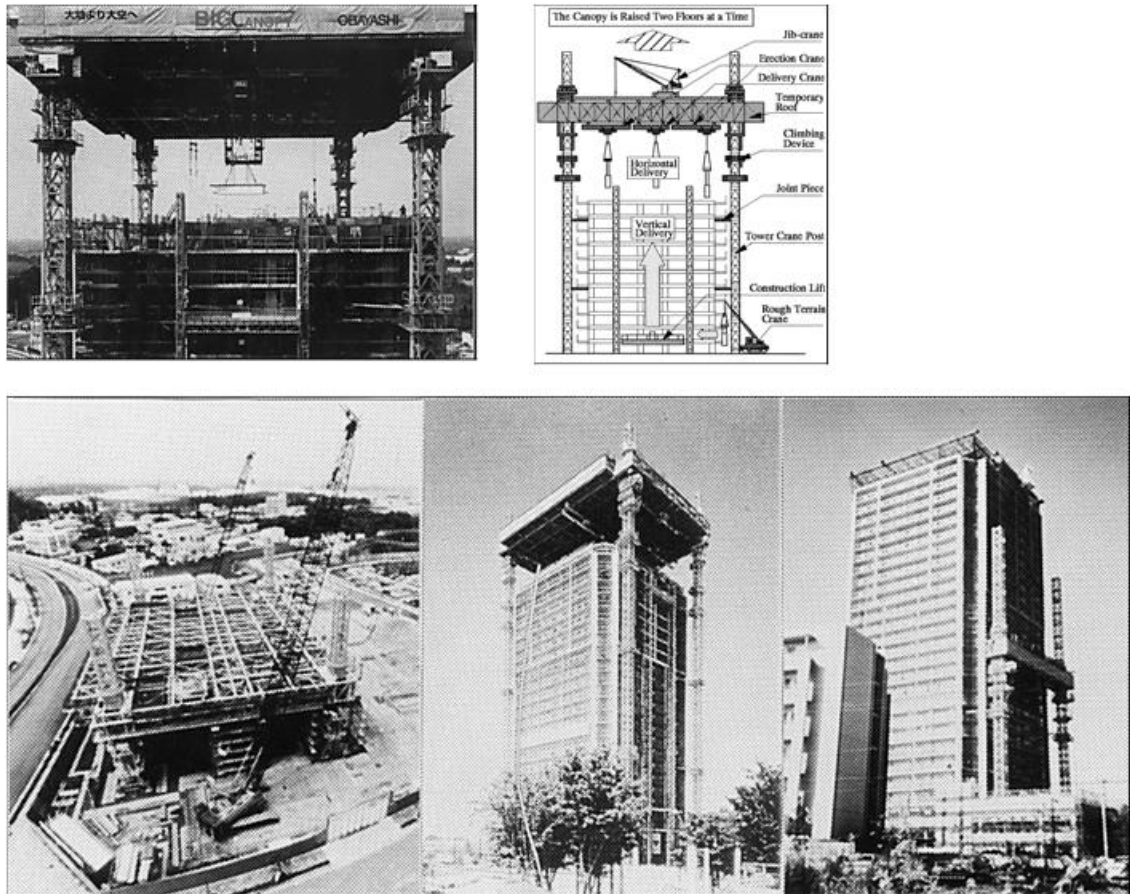


Figure 2.9 – ‘Big Canopy’ by Obayashi Corporation (Shiokawa et al., 2000).

One critical limitation in the construction imposed by the systems was their inability to adapt to various differentiations of site conditions and programmatic customized variations in their productions. They can manufacture identical structures in an extremely efficient manner; however, the flexibility of the systems was not ready to be used at the

practical level for all kinds of project types. Some companies employ hybrid applications, with a conventional erection method for the portions of buildings to sustain the developed technologies and investments they have committed. Overall conceptions were extremely ambitious and promising, though the concept of construction automation might have needed another level of conceptual refinement or theoretical leap for the systems.

Assembler-Assemblee Relationship

Whether a system has a clear separation between what is going to be built and what is building it is one criterion that can classify the system's characteristics. This separation between assembler and assemblee consequently leads the system to have discrete, separate phases of design and construction. If any system's composing units have an ability to construct themselves, the system can more spontaneously reconstruct itself without having any clear termination points for each phase. The construction automation systems still show a clear separation between assembler and assemblee, although their systems are fairly automated. As long as this separation exists, no matter how automated the systems are, buildings will not be able to actively and dynamically reconstruct themselves. Design and construction are two discrete, separate phases in the construction automation systems, and they still fall into the aforementioned closed-planning category. Realization of time-based growth models in architecture might require seamless integration between those two roles – assembler and assemblee.

The distinctions between buildings and their fabrication systems may fade away in the future, and the system of production can manifest itself as an inseparable component of architecture in order for structures to adapt and grow over time. This ultimate integration

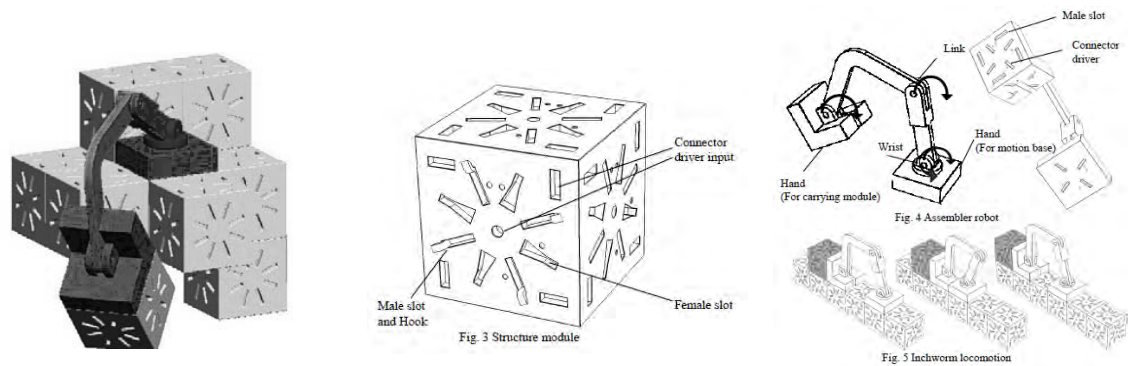
of production systems and architectural spaces will realize a true sense of ‘architectural automation’ which includes automation of not only construction but also design. This *generative* fabrication machine is simultaneously a machine for living that can also spontaneously think, design, and build. We have yet to see the realization of such systems in architecture, and the physical appearances of such systems may be entirely different from our conventional understanding of architectural forms. The realization of the system will introduce a different kind of formal appearance for future structures.

Recent advances in technologies allow us to reconsider the notion of fundamental composing building units, from being inert objects to being active responsive units. This will alter the building units into active intelligent components that can respond and reconfigure to fit unpredictable future scenarios. In the next section, I would like to review more specific potential applications to embody these hypotheses.

2.2.6 Swarm Intelligence: Self-reconfigurable Robots

Precedents from a more speculative area are the modular systems proposed by computer scientists such as Murata (2006). Such modular systems can self-reconfigure themselves to produce structures such as dams, caissons, bridges, and space structures. Metabolist architecture can also be categorized as a modular system; the critical difference between Metabolist architecture and self-reconfigurable systems is that the modular units in reconfigurable systems have abilities to sense and evaluate the local conditions and autonomously take proper actions. In this way, the systems have an ability to function and recover even with some failures at several local conditions, while centrally controlled

systems fail entirely once the main controlling units stop working. This embedding of a local feedback system within each individual subunit enables robust control of adaptable structures. In this regard, recent implementations by computer scientists are more advanced interpretations of biological metabolic systems by being an aggregation of automata as a reduced form of living cells. This adaptation of another level of intelligence within the individual units of the structure starts finding practical applications in construction in hazardous areas or construction of unmanned planetary structures where we cannot depend on any assumptions of a pre-determined environment.



Figures 2.10 – Self-Reconfigurable Modular System from Nomura et al. 2006.



Figure 2.11 – ‘Self-Reproducing Machine’ from Lipson et al. (2000).

Mobile robot motion planning can be classified into two categories: centralized and distributed. Certainly the distributed algorithms are used for the control of swarm robots,

and these are the important notions for the self-reconfigurable systems. Regarding the existing control systems of architecture, components of intelligent building systems, such as heating, cooling, and lighting, can have distributed controls instead of centralized ones. Local sensing, control, and actuation can be done in a distributed manner all around the buildings, and they may provide autonomous systems that can respond to unpredictable changes in their environments without central control by supervising units.

A more extreme conception of a modular system can be a ‘Smart Dust’ system as introduced by Warneke et al. (2001). Simply reducing the size of the fundamental composing mobile units down to the scale of ‘dust’ will have potential to introduce a new kind of flexibility into our architectural envelopes. In biomedical application fields, the ARES (Assembling Reconfigurable Endoluminal Surgical system) project from Italy is envisioning the use of modular robotic systems that can be inserted into a human body and can configure themselves into kinematic structures to operate inside the body (ARES, 2007). Potential applications of such concepts can be glazing systems with millions of apertures in nano-scales which can be adjustable based upon the temperature, humidity, and light. A more radical thought would be a building composed of tiny cells that can respond to environmental changes, and can grow into morphologically desired configurations. Furthermore, the actual subunits of the system can be made out of organic biological materials or hybrid materials between synthetic materials. Finer and more universal subunits can potentially adopt more varieties of tasks without having preprogrammed knowledge of the possible work, and this flexibility is one of the greatest potentials of the systems based on the swarm logic.

Recently, use of RFID tags for material identification is getting attention from many people in construction industries. Use of the smart dust devices (Werneke et al., 2001) can surely provide the information simply beyond being identification tags: measuring materials' wear, detecting conditions of fatigue and so on. Furthermore, sensing and actuation capabilities in micro-scale motes will allow desirable air-flow and heat releasing mechanisms in the building envelopes. Kinematic abilities within the individual motes (units) can adjust tiny vents for air circulation. Aggregations of movable cells that can compose building skins into morphologically desired configurations to satisfy the concurrently changing needs may be realizable in the future. Inability of buildings to adapt and grow according to changes in environments and needs over time is a dilemma for contemporary architects, and the concept of the micro-scale devices and further development of them might give us the vehicle to solve these problems.

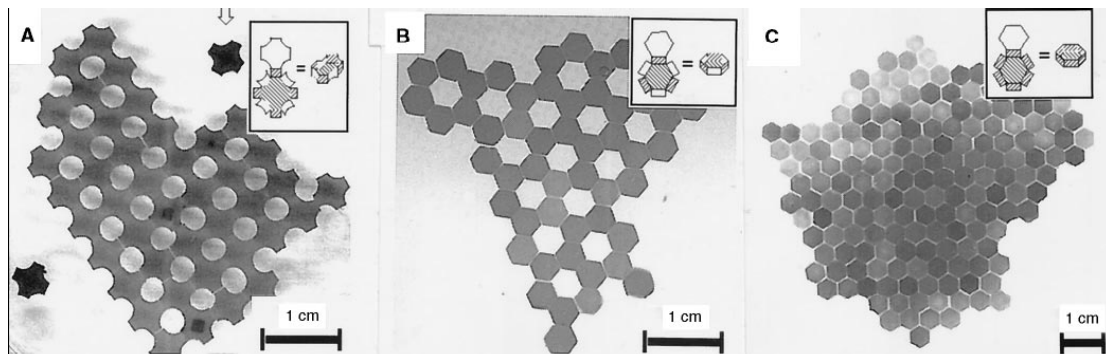


Figure 2.12 – Self-assembly results from minimization of the interfacial free energy of the liquid-liquid interface (Bowden et al, 1997).

Additionally, one precedent that represents the issues of sizes of the elementary subunits of systems is introduced by Bowden et al. (1997) in their paper, “Self-Assembly of mesoscale objects into ordered two-dimensional arrays.” This paper introduces

‘artificially’ simulated examples of self-assembly and proves that the local interactions of synthetically fabricated simple components can produce global configurations without the presence of any external supervisions or directives. The experiment uses aggregations of individual unit objects made of polydimethylsiloxane placed inside an oscillating rotary shaker. Each object’s surfaces have different ‘*wettability*’ – either hydrophobic or hydrophilic – and the objects interact at the interface of water by lateral capillary forces. The author claimed that ‘*Self-assembly results from minimization of the interfacial free energy of the liquid-liquid interface.*’ This technique is aiming for the fabrication of complex systems such as applications in microelectronics, optics, micromechanical systems, and displays, and the paper explains that the fairly simple characteristics of individual members can achieve an emergent formation of cohesive structures by interacting with each other. This experiment may not have a direct link to architecture in its scale and material choices, but it is one of a few examples that represent the notion of generative fabrication by artificial physical objects.

2.2.7 Self-Replicating machines: 3D printers

In the previous section, I explained that the construction automation during the 1980s in Japan suggested the potential concept of self-reproduction in architecture. However, there was still clear separation between assembler and assemblee, and there was no integration between them. Within the contemporary practices in computer science, there exist more progressive realizations of the self-reproductivity concept. The RepRap project by Adrian Bowyer at Mechanical Engineering at the University of Bath and the Fab@Home project by Hod Lipson at Cornell University are examples (2010). Bowyer explains self-

replication as the ability to reproduce components that are necessary to build a child version of its own replication (RepRap, 2010). The RepRap system is a 3D printer that builds parts up in layers of plastic, and the system has successfully printed majorities of its own components to construct its child and grandchild models. Bowyer argues that the system can conceptually demonstrate evolution and increase in number exponentially. As we reviewed in earlier sections, many natural systems possess not only the ability to self-reconfigure but also the ability to self-reproduce its components, similarly to cell growth processes in any biological system.

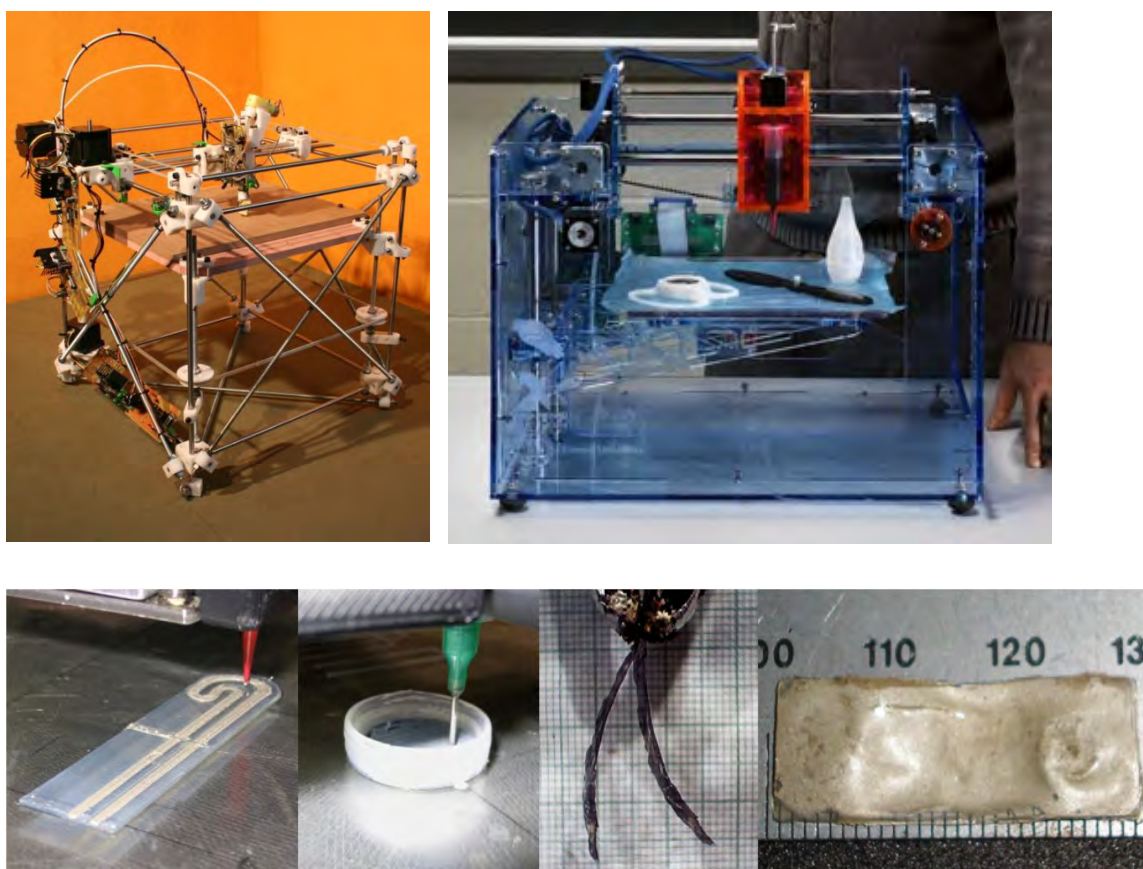


Figure 2.13 – RepRap (left) from <http://reprap.org/bin/view/Main/WebHome> (RepRap, 2010). Fab@Home, (right) Portable 3D printer by Lipson (Malone and Lipson, 2006). Printable Battery: freeform fabrication of a complete Zn-air battery (bottom-left) Printable Actuator: Ionomeric Polymer-Metal Composite artificial muscle (bottom-right) by Lipson (Malone and Lipson, 2005)

These speculative research projects by computer scientists represent a great endeavor to realize artificial emergence. We reviewed self-reconfigurability seen in swarm robotics using distributed controls, and the concept of self-reproduction or self-repair seen in current digital fabrication systems. They both seem to represent great steps for artificially creating a subunit that can assimilate emergent growth processes over time. Self-reconfigurability, distributed controls, and a concept of self-reproduction or self-repair are definitely key characteristics for artificially creating a subunit that can assimilate emergent growth processes over time.

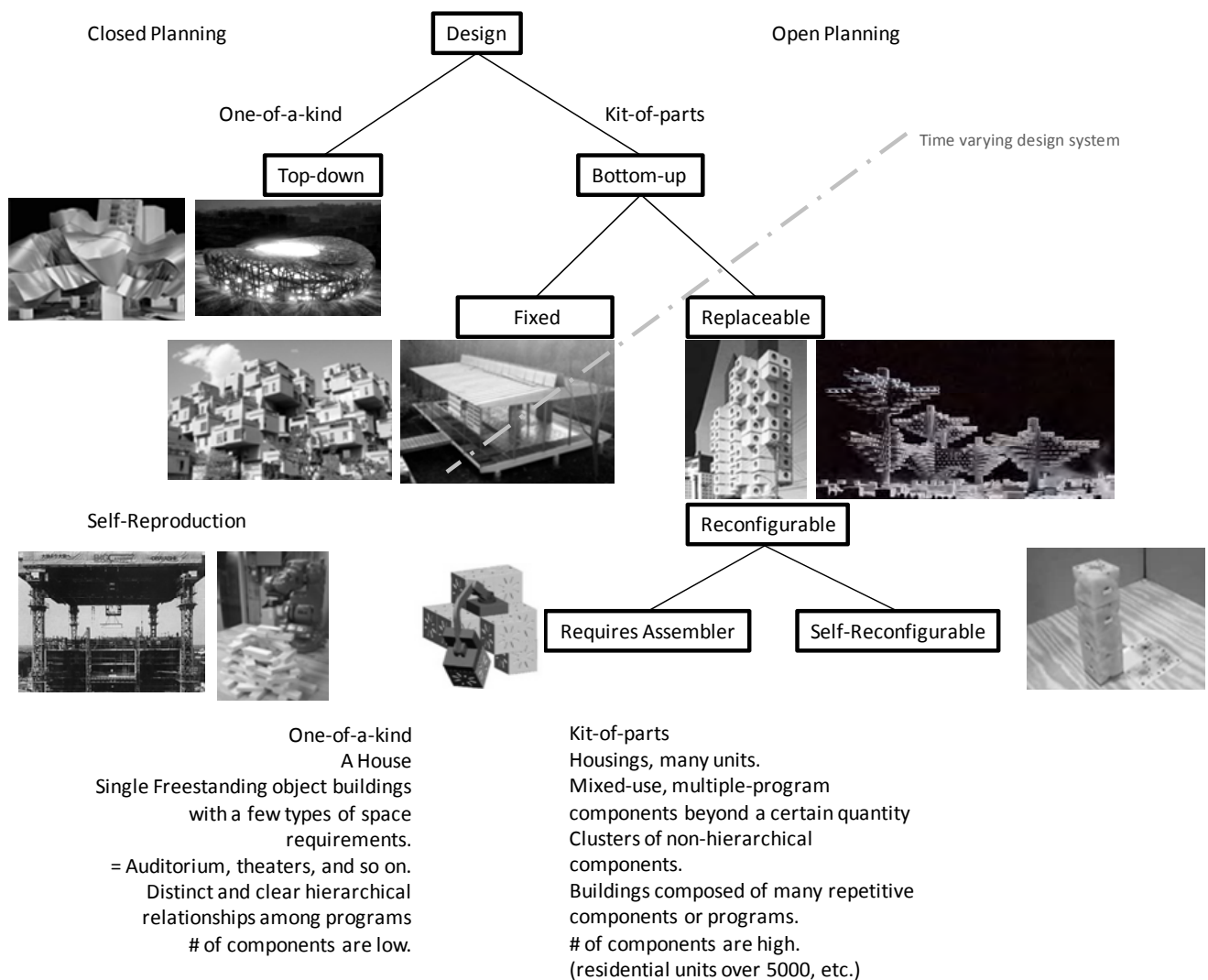


Figure 2.14 – The Matrix that shows various systems. Categorized based on the concept of open/closed planning.

2.3 Urban-Scale Systems: City Formation and Emergent Growth

The following section reviews self-organizing characteristics seen in many urban-scale design examples. The examples from urban scale are often results of collective decisions over a long span of time. Informal settlements are one of the classic examples of emergent formation in urban scale. I will also review designers' active efforts to integrate group dynamics into design developments. After the review of actual realized precedents, I would like to examine virtual systems that simulate growth in or production of cities. Together with physical and virtual precedents, this section will introduce real urban phenomena and their potential procedural descriptions.

2.3.1 Kowloon Walled City – Architecture in a state of anarchy

Ironically, one of the man-made structures that display the dynamics of self-organization was not planned by any notable individuals or groups of architects. Kowloon Walled City in Hong Kong (Lambot, 1999) was neither a beautiful nor successful living space in our standard sense of perception; however, this is one of the rare synthetic examples of emergence in a building construction process which was achieved within less than the lifetime of a single generation.

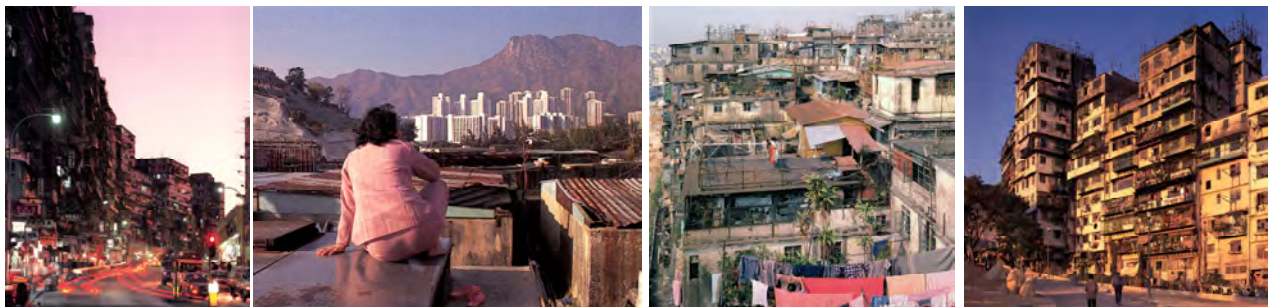


Figure 2.15 – Kowloon Walled City in the 1980s before its demolition (Lambot, 1999)



Figure 2.16a: Kowloon Walled City in 1973 (Lambot, 1999)



Figure 2.16b: Kowloon Walled City in 1994 before its demolition

The political relations between China and Britain during the last century had made the Kowloon area in Hong King a political and diplomatic black hole which had been free of any laws or regulations from either government. As a consequence, Kowloon had become an asylum for many refugees escaping from the civil war and political prosecutions by the Communist government at the time, and had formed a rare model of an anarchist city.



Figure 2.17a: South Elevation of Kowloon Walled City (Lambot, 1999)



Figure 2.17b: Plan of Kowloon Walled City (Lambot, 1999)

There were only two regulations. One was a fourteen-story height restriction due to the proximity of Kai Tak Airport, Hong Kong's old international airport. The other was the authorized installation of electricity to reduce the risk of fire from open flames.

The structure was not planned from the outset for residents numbering more than 35,000; they had displayed incredible tolerance and adaptability in accepting radical increase of their population over a relatively short period of time. The city might not have been the most efficient building in terms of circulation. Labyrinthine interior networks were far from efficient compared to ideals of conventional axial building planning in modern architecture planning. However, each resident's needs for basic living were mostly satisfied to some degree by constantly altering its morphology. In other words, scarcity in some functionality had triggered motivation for another new construction to adapt new features, such that the complex as a whole displayed incredible robustness and flexibility.

We have witnessed self-organizing qualities in the formation of many historical cities, and cities such as Mouray Idriss in Morocco are often referred to as an example of emergence in human civilization. But the Kowloon example had been accomplished within a relatively short period of time and on a scale of several building complexes. This self-organizing structure actively adapted and regenerated itself into a working model of a metabolic system in architecture by accommodating endless incoming refugees. Unprecedented adaptability exhibited in this project was achieved by relatively unsophisticated and nonprofessional carpentry work by local residents using low-quality reinforced concrete construction.

The crudeness in construction qualities contributed to the spontaneous growth of the structure to some degree. In exchange for construction precision and quality, they gained extremely unique, almost organic, plasticity in erection sequences, and these seemingly primitive construction methods led them to achieve unprecedented adaptability in their architecture. Not only Kowloon Walled City, but many temporary housings or shelters in many post-war sites in the world have often possessed a similar organic quality. They are often called “barracks” and are usually made of debris.

The residents sacrificed elements of efficiency by continual additions and seemingly haphazard extensions and alterations. In return, they resolved many individual needs among local neighborhoods and were almost heuristically grown into a multi-programmed complex structure which included schools, kindergartens, shops, hospital, dentist, and communal spaces such as courtyards in many residual spaces.

Urban Growth of Cities

Mouray Idriss, etc



Growth periods in Years:

100 -1000 years

Primary composing materials:

Masonry

Boundary conditions (constraints):

geographical + defense

Traditional Settlements

Yemen, Ethiopia, etc



Over 1000 years

Mud-Bricks

geographical + defense

Building Scale Application

ex. Kowloon Walled City.



50 years

Reinforced Concrete

+ strong boundary conditions

Informal settlements

Favela in Brazil



5 – 20 years

Debris, Scraps, Masonry, etc

+ constraints from city zoning

Figure 2.18 – Different types of urban growth models

Of course there were many negative aspects in Kowloon Walled City, such as absence of natural lighting deep inside the structure. Some labyrinthine corridors needed to be constantly illuminated by fluorescent lights even the middle of the day, and the same construction or design methodology cannot be directly applied to our contemporary social environments. It is unlikely that average people of the day will be willing to live in this chaotic stew of social entities. However, I believe that the type of growth model that Kowloon represented showed us a potential existence of completely different methodologies in building design compared to our relatively top-down conventional design strategies. We may find some potential benefits by learning from fundamental design principles unique to Kowloon Walled City.

2.3.2 Participatory Design Guided by professionals

In contrast to the physical implementations that we have reviewed in the last section, there are strategies to use group dynamics directed by human participation to evolve designs. Lucian Kroll and Christopher Alexander are well-known for their participatory design practices. While many practitioners, such as Metabolists, had emphasized the invention of new physical components, Kroll and Alexander's strategies seem to put more emphasis on user involvement in design processes. Just as some of the synthetic systems in various areas have started to gain benefits by adopting the distributed logics inspired by many natural systems, Kroll and Alexander believe that architectural design

processes can potentially benefit from the application of decentralized thinking as a core ideological innovation.

Alexander introduced “a pattern language” in 1977, proposing 253 unitary patterns derived from traditional architecture. These patterns can be used as a generative grammar allowing participation of non-professionals and users to open up the processes of design. Later, his concept of the pattern language has become an inspiration for the current concept of object-oriented programming beyond the field of architecture. An idea of a production platform that allows multiple users’ engagement has become an inspiration for many object-oriented programming languages. They allow multiple programmers’ simultaneous participation to achieve a common goal for their production. This approach has become a key concept for the foundation of modern software engineering, accelerating the production rate. However, the spatial and geometrical production platforms of such systems are yet to be clearly defined in many fields. Some CAD systems allow users to work on different areas of building plans by external references, but they are merely for drafting and documentation purposes and not useful for organizing to reflect and to integrate multiple designers’ feedbacks into a production done in a collective manner. There are not many successful computational multi-participatory production platforms that can support design processes of spatial and geometrical forms. Although Alexander’s endeavors are still based on manual processes without use of computers, they are one of only a few examples that challenge the notion of cooperation in the context of multi-user engagement platform development for physical spatial design production.

Lucian Kroll's work is often called *anarchitecture* due to his unique design process based on group dynamics (Gassel, 1979). His work displays a more or less similar quality to what we have seen in the Kowloon Walled City. In his housing project for medical students in the Catholic University in Louvain, he intentionally avoided being an authoritative expert figure. Rather than imposing a top-down master plan, he stayed as an *organizer* throughout the entire design and construction process. In order to gain maximum feedback and ideas from his clients, Kroll believed in group participation. He objected to the standard role of the architect who often imposes his or her own ideal and aesthetic values.

In this project, students as the future users negotiated with each other to design their own living spaces. In the beginning of the process, he provided a large open space for a group of students. Then the interior space was divided into smaller cells by industrial modular wall systems. Walls were installed by the students themselves. The resulting plan does not possess any clear global organizational patterns like those seen in conventional academic facilities; they simply look quite random. In fact, this is a rare example of design that was initiated and directed primarily from agglomeration of local-level negotiations. As a result, priority for design decisions was given to the satisfactory conclusion of the local-scale environments over imposition of global scale functionality. It was a unique experiment to seek global order from many local interactions.

The same disorderly quality dictates the appearance of the buildings and their façade. No two window fenestration patterns look alike. It was partly due to Kroll's ideology of avoiding mindless repetitions. He intentionally used this strategy to gain maximum

diversity in his design. It is a question whether the strategy has any connection with his more profound attitude toward participatory design, but the Kowloon-like disorderly appearance of his design may have some relevance to his unique derivation of building planning based on group dynamics.



Figure 2.19 – Medical Student Housing at Catholic University of Louvain by L. Kroll (left)
Interior Space made freely by students using panel wall system (middle & right) (Gassel, 1979)

Just as some network communication applications are gaining adaptability and robustness within their systems by applying decentralized network hubs and connections, applications of bottom-up logic in design may have a potential to introduce new adaptive qualities in building. Kowloon Walled City in Hong Kong had displayed incredible flexibility and capacities to transform itself into a structure to accommodate more than 35,000 inhabitants from a few thousands within a relatively short period of forty years. The resulting structures had not produced all positive consequences in terms of many aspects of living such as lighting, circulation, and hygiene; however, the flexibility exhibited by the structures lends credibility to the existence of building processes different from conventional design methodologies typically led by professionals with comprehensive blueprints. Self-organizing characteristics seen in these precedents show some unusual flexibility, and active applications of computation may enhance them by compensating for the aforementioned shortcomings.

Review of Urban-Scale Simulations: Computational methods

There are existing software applications to generate hypothetical virtual cities. In the next section, I would like to review several of them and explain their different characteristics in processes.

2.3.3 Fractals, DLA, and Agents: Examples by M. Batty at CASA

One of the earlier attempts to describe growth processes of cities mathematically and scientifically was made by M. Batty at the Centre for Advanced Spatial Analysis (CASA) at University College London (UCL). In *Fractal Cities* (1994), he used fractals as a mathematical model to simulate urban settlement patterns of various cities and found correlations among his mathematical models and the growth patterns of actual cities. His work has focused on analyzing correlations among urban densities and spatial and geometrical characteristics of urban growth. He has also looked at algorithms such as Diffusion-limited aggregation (DLA) and has provided potential new approaches to measuring urban densities. His study has become one of the pioneering works in the computational city modeling field and has established several potential application areas for computational methods such as DLA and computational agents.

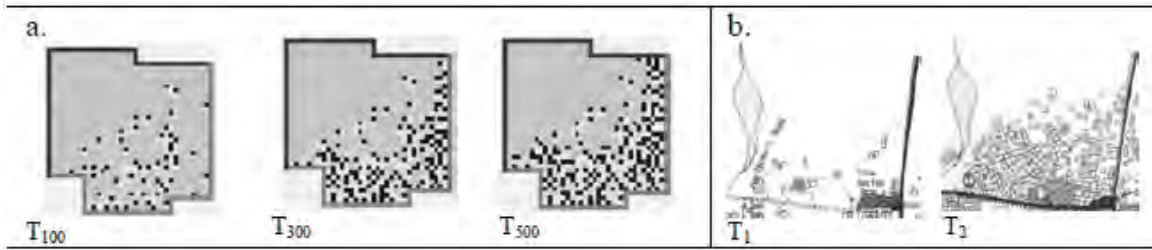


Figure 2.20 – Sequential output of favela experiments (Left). Urban growth simulation of Acera, Ghana using agents by Sobreira at CASA, UCL (Right). (Barros, J. and Sobreira, F. 2002)

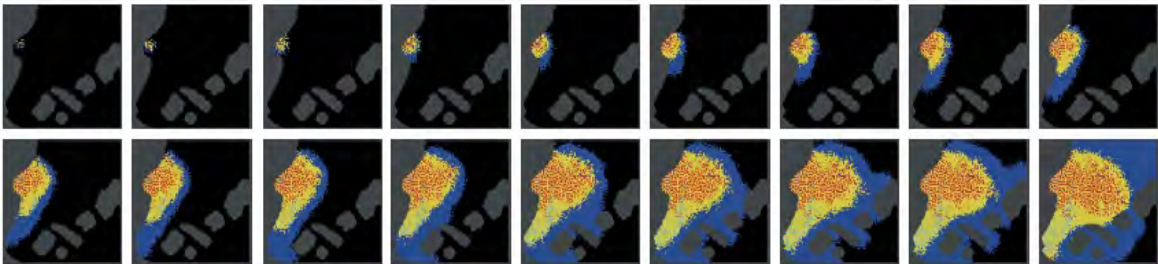


Figure 2.21 – Urban growth simulation of Porto Alegre using agents by Barros at CASA, UCL. (Barros, J. and Sobreira, F. 2002)

Batty's work had mainly focused on relatively formally developed modern cities such as London, well into the mid-1990s. However, in 2002 Barros and Sobreira from CASA at UCL, under the direction of Batty, introduced a model to simulate spontaneous growth of favelas in Brazil using a decentralized agent model. A decentralized agent model can simulate collective actions by multiple individuals, and enables the system to describe gradual growth. Their outputs are based on a discrete cell environment and do not provide as many geometrical details of building information as the model by Muller. However, the use of computational agents is a valid strategy enabling Barros and Sobreira to implement local interactions in time series. In their system, agents do not use any natural/environmental conditions as an attractor for their behavior. There are also similar systems called 4D-CAD systems developed by research institutes in many general construction companies, but they are primarily for organizing on-site construction sequences in time series.

The development of a novel system that can simulate the spontaneous growth of cities in precise geometrical detail based on a given landform and environmental conditions will be a contribution to both landscape and architectural computation studies. In this thesis, experiments in Chapter 6 present a different approach to this goal.

2.3.4 L-system and Shape Grammar: City Engine

Müller and Parish from ETH Zürich have developed a system called CityEngine (2001) which can create realistic representations of comprehensive 3D city models including roadmaps and buildings from a few sets of statistical and geographical input data. The system is intended to produce outputs highly controllable by users. The system's realistic outputs have been used for production of virtual movie scenes.

However, their system is not designed to simulate the gradual growth processes of cities and hence cannot represent spontaneous settlement patterns over time. Müller's generative process relies on existing cities' knowledge-based typological patterns in order to provide one particular state of a city, but not a continuum: the rectangular grid in New York, the Haussmannian radial grid in Paris, or the residential development of an American suburb. The system also uses correlation between landscape and building layout as a relatively minor criterion for generation of the city. Thus, the system's single-shot outcomes are not a simulation of emergent processes.




Pattern name	Pattern	Example
Basic	No superimposed pattern.	
New York	Rectangular Raster	
Paris	Radial to center	
San Francisco	Elevation min or max	see figure 6

Table 1: An overview of the street pattern used in the CityEngine system with a short description and an example.

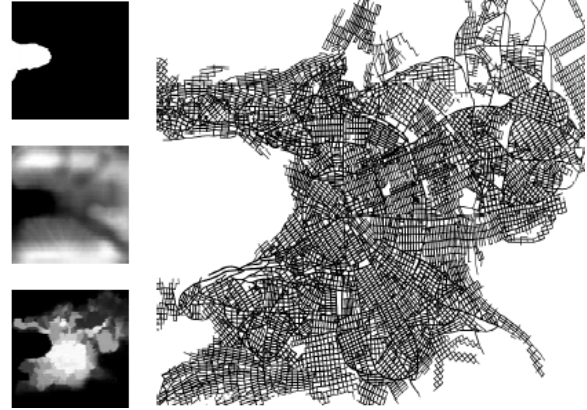


Figure 2.22 – Street Patterns used in Procedural Modeling, CityEngine. (Müller and Parish, 2001)

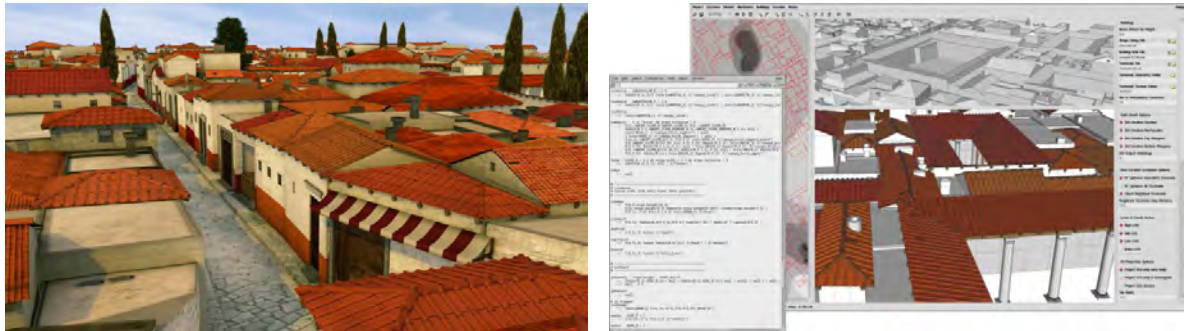


Figure 2.23 – CityEngine (Müller and Parish, 2001)

Their generative method is based on a procedural creation of virtual urban environments. This procedural approach is also often called grammar-based generation of models and has been used widely in computer graphics to create plant geometries. Lindenmayer originally developed grammar-based descriptions of plant geometries, and this method is called L-systems after him. In L-systems, geometries are represented and encoded in sequences of strings, and the geometric creations and transformations are replaced by directly operating on these symbolic representations of strings. These operations become rules for grammar of a certain geometric group. Later, in the field of architecture, G. Stiny developed “shape grammar,” which defines rules directly on shapes instead of operating on strings. This shape-based symbolic transformation has been used by some

communities of architects and designers to develop design grammars that can capture specific styles of design instances or signature expressions by famous architects such as Palladio, Frank Lloyd Wright, Alvaro Siza, and others. This system captures some design characteristics of individual expressions quite well and has produced some instances that are almost indistinguishable from originals (Duarte, 2002).

These transformation rules need to be obtained, defined, or formulated somehow, and there is no automatic, or even systematic, way to automatically create these rules. There are some attempts to auto-generate these rules with the help of evolutionary computation techniques such as genetic algorithms. Combining a genetic algorithm and the shape grammar has been proposed to heuristically find rules that satisfy fitness factors. However, this approach is computationally expensive and not practical at the level of extracting the rules that define signature expressions of any actual human designers.

Müller's system was also based on this shape grammar and L-system, and the choice of these methods was appropriate for capturing certain characteristics of city street and building layouts. However, those characteristics are originally imposed by the creators (or users) of the system as transformation rules, and the outputs of the system are, to some degree, influenced by their interpretations of city layout. Their intents are to produce comprehensive visual outputs of a city using a relatively small set of statistical and geographical input data, but achieving the fabrication of fine details without providing any knowledge-based interpretations from existing precedents is not yet even close to being feasible. Usually the results from these systems allow some level of reliance on data external to the sole input information.

Unlike extremely scientific and analytical attitudes toward derivation processes of output designs by researchers such as M. Batty at UCL, Müller's priority seems to be the visualization of detailed geometries. But, as a representation tool, there is no doubt that the tool performs extremely successfully. The design method proposed by Alexander and Manheim (1962) nearly four decades ago displayed a quite contrasting approach that does not use any data external to the subjects under the design, and I will review this in the next section.

Müller's system from 2001 produces only an urban model that is static in time, but a new paper (Müller et al. 2009) represents a system that can simulate an urban model over time. In addition to L-system and shape grammar from their previous system, they include growth and expansion rates for streets and some building blocks. Those rates are influenced by various user input parameters that control traffic intensities and economies. Again, in this system, priority was given to the generation and control of realistic and detailed three-dimensional models, and many geometrical configurations are given by user inputs or Müller's implementation of city and building forms.

This relatively top-down characteristic of their generative system is probably suited for simulating highly detailed models of cities which were produced as a result of imposition of modern city planning methodologies. Paris, Brasilia, and New Delhi probably are good examples of those cities, being highly influenced by modern urban planning concepts such as zoning. However, simulation of informal and spontaneous settlements over long periods of time might be a challenge. Oftentimes urban configurations developed by

spontaneous settlements are not the results of impositions of specific urban forms and are hard to represent by a collage of discretized typologies.

In conclusion, a system such as CityEngine can simulate urban models that are the product of modern planning methodologies based on relatively top-down planning, and Müller has proved that the system can potentially simulate growth over time. However, simulation of informal (spontaneous) settlements may require a different type of methodology, as the characteristics of these settlements were normally not imposed (produced) by any hierarchical structure of typological templates based on procedural operations. Alexander offered a similar discussion in his essay, “A City is not a Tree” (1965) and referred to two different types of city formation and planning processes.

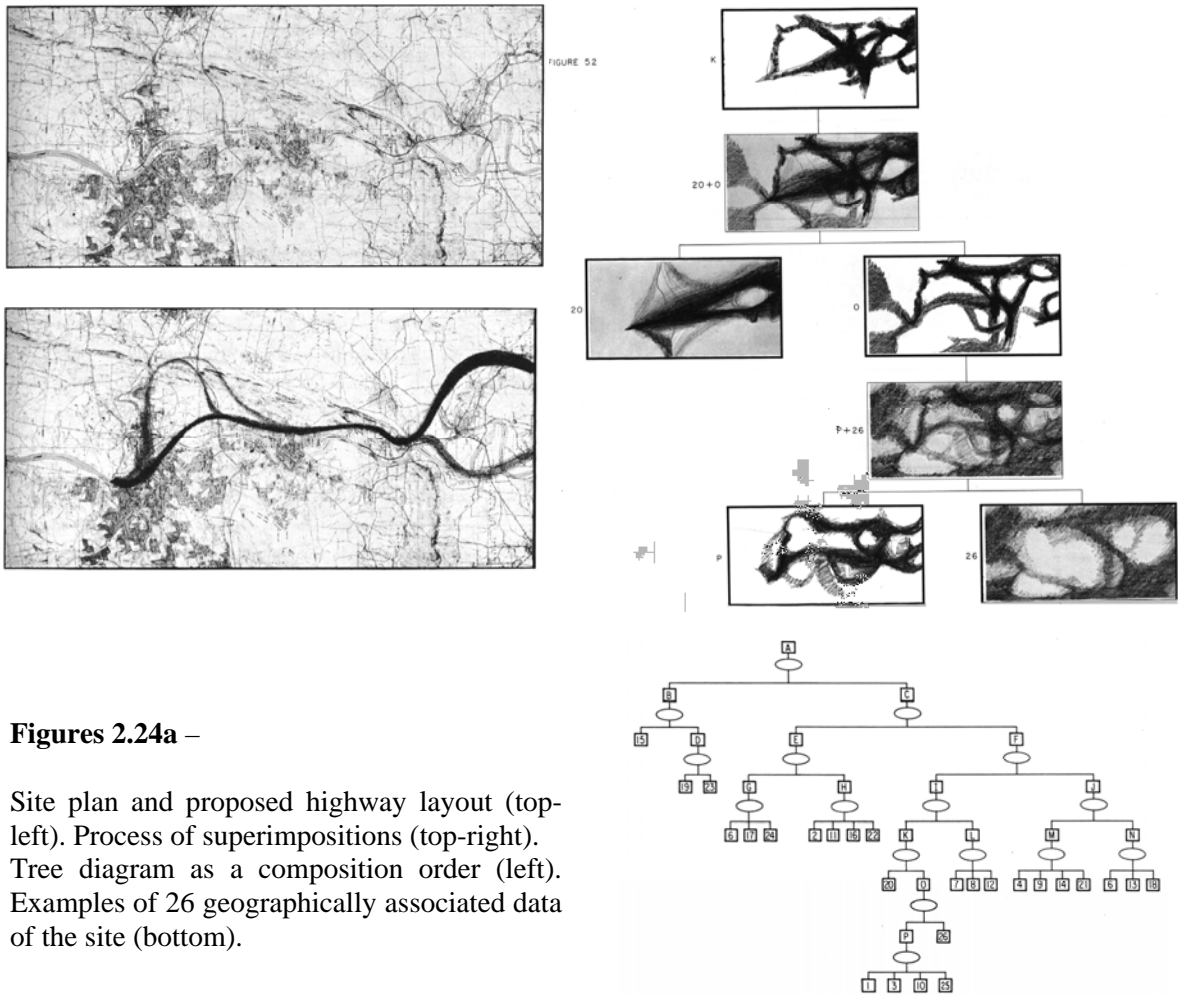
I want to call those cities which have arisen more or less spontaneously over many, many years natural cities. And I shall call those cities and parts of cities which have been deliberately created by designers and planners artificial cities. Siena, Liverpool, Kyoto, Manhattan are examples of natural cities. Levittown, Chandigarh, and the British New Towns are examples of artificial cities.

It is more and more widely recognized today that there is some essential ingredient missing from artificial cities. When compared with ancient cities that have acquired the patina of life, our modern attempts to create cities artificially are, from a human point of view, entirely unsuccessful.

Alexander’s criticism about the modern planning methodologies makes us aware that there is a different possibility, for different implementation of a city generation process.

2.3.5 Superimposition Technique by C. Alexander and M. L. Manheim

Alexander and Manheim in "The Use of Diagrams in Highway Route Location" (Alexander and Manheim 1962) proposed a design derivation method that is different from methods based on symbolic design patterns and transformations, often represented by shape grammar.



Figures 2.24a –

Site plan and proposed highway layout (top-left). Process of superimpositions (top-right). Tree diagram as a composition order (left). Examples of 26 geographically associated data of the site (bottom).

Their method does not impose any formal design patterns in physical format, yet their system shows efforts to extract design solutions that are fairly true to input data. This design derivation process seems to possess similar characteristics to those of self-

organization seen in many natural systems. New design of a road in a landscape emerges from correlations among 26 geographically associated data of the site, such as land costs, earthwork costs, noise level, travel time, drainage patterns, air pollution, and so on. These 26 data maps show different degrees of suitability by shade of gray with respect to the aforementioned different criteria or parameters. They overlay and superimpose these gradient maps based on a specific grouping order and weights for each map using a composition proposed in their tree diagram. The result of the composite of factors produces a spatial intersection of feasible areas for a proposed highway lines with darker tones representing more desirable areas.

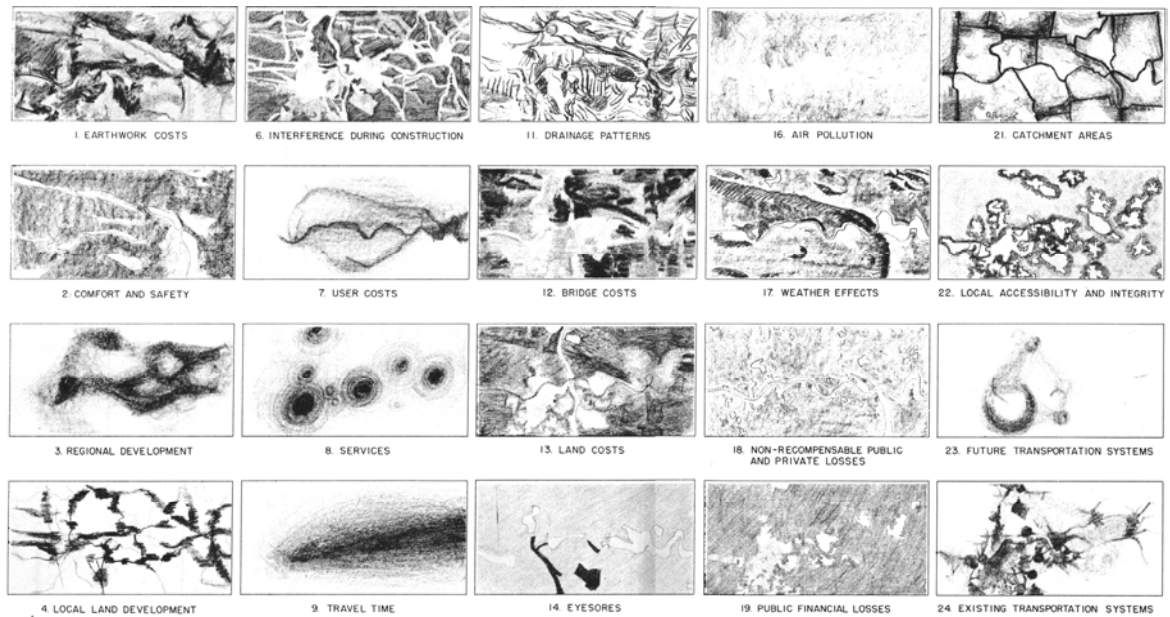


Figure 2.24b – From “The use of diagrams in highway route location: An experiment” C. Alexander and M. L. Manheim (1962).

In their experiment in the 1960s, superimpositions of gradient maps were all executed manually and seem to lack precision in the process. However, the novel derivation process of the method is obviously the most valuable part of the study, and we now can

simulate the same superimposition technique using readily available graphic processing applications and data from Geographic Information Systems (GIS). Derivations of physical design configurations without providing any hints of predefined formal design templates are extremely rare in any past design methodologies, and this short study sheds light on design derivation principles similar to characteristics seen in self-organization and emergence.

The only relatively unconvincing part of the experiment is the way Alexander and Manheim introduced the tree diagram as a composition order. This tree seems to be relatively arbitrarily imposed by them, and there is no sufficiently logical explanation why they chose this specific composition out of all possible orders. This formulation of a tree diagram can be gained from correlations among examples of similar highway designs and their conditions from many existing sites. For example, use of learning algorithms such as neural networks to develop a tree-like structure solely from data from many existing highway structures would be ideal. We can form a neural network based on ample data and resulting road configurations from many existing precedents and derive a tree-like diagram using a back-propagation technique solely from given data without any subjective interpretations. This thorough approach could be more appropriate for the original intention and ambition of their project.

Additionally, the superimposition of gradient maps is all done by linear summation of pixels' tone value. This over-simplified process may fail to capture some inter-relational characteristics among data and road configurations. A good example similarly indicating this shortcoming can be seen in a comparison of a single-layer perceptron (feed-forward

neural network) and a multi-layer perceptron. The single-layer perceptron is also known as a linear classifier, and it can only solve linearly separable problems. In general, two point sets are linearly separable in n -dimensional space if they can be separated by a single decision surface (a topologically $(n-1)$ -dimensional hyperplane). For example, a logical exclusive-OR (XOR) function's two-input patterns cannot be separable with a single line (Minsky and Papert, 1972). On the other hand, the multi-layer perceptron can solve linearly non-separable problems using the back-propagation technique. Here, we are seeking the unknown classifier that can provide feasible highway locations from various data inputs. Since we do not know whether this unknown classifier will be dealing with linearly separable or non-separable problems, it is preferable to use techniques that have the characteristic of a multi-layer perceptron.

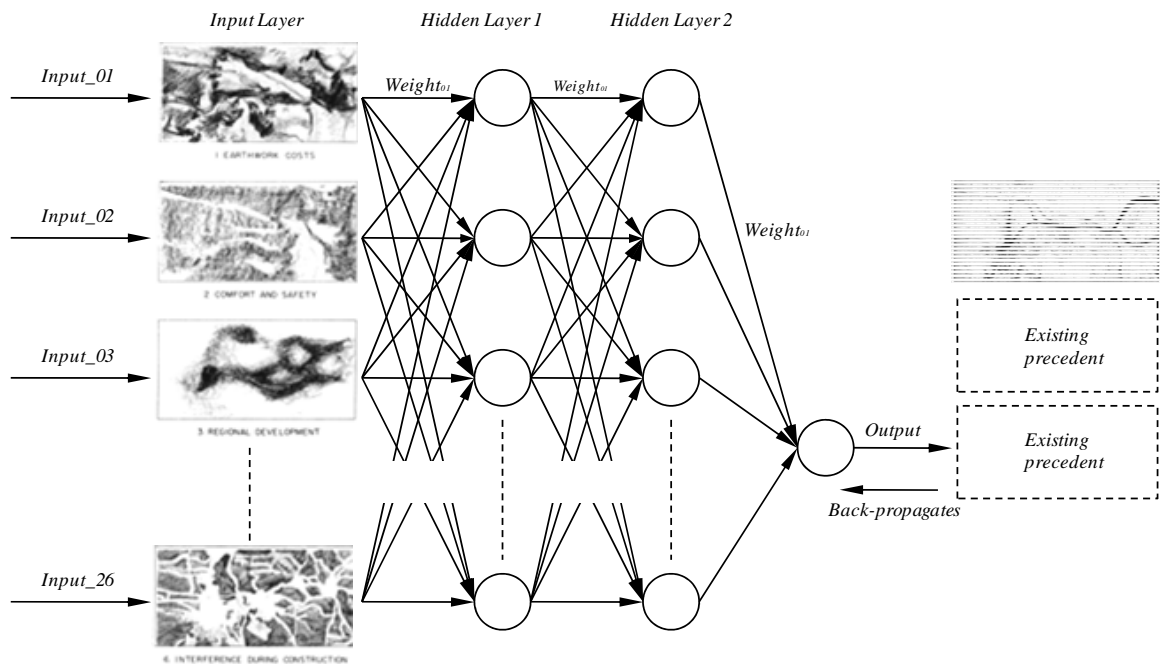


Figure 2.25 – Conceptual diagram showing a Neural Network with 26 inputs.

Another important question is how we can extend this method to real-time dynamic scenarios. Many criteria used in this experiment, such as local population density, travel time, and various costs associated with geographic locations, are time-varying dynamic values. In principle, those gradient maps can be expressed as a function of time, and solutions from those criteria can also be a non-static ever-changing figure. The increasing processing power of computers allows us to handle dynamic data, and the type of design derivation described in this experiment will have more promise with the use of computation.

2.4 Summary and analysis of the background

In the previous sections, natural systems and artificial systems are reviewed in regard to emergence and their self-organizing characteristics. In the following section, as a conclusion of this chapter, I would like to summarize and classify the types of systems that were reviewed in the previous sections. In order to compare and evaluate different natural and artificial systems, Time and Scale are key criteria. With respect to time and scale, I would like to analyze what is missing in current artificial systems compared to the examples of natural systems, and speculate about future directions for improvement of current artificial systems.

2.4.1 Subunits

All the systems introduced in the last section are composed of some kinds of subunits. A whole system consists of an aggregation of finite elements with certain properties and/or active behaviors. We learned that local interactions among these elements are normally the main directives to create globally meaningful behaviors. These subunits can be inert or animated, static or self-driven objects. Sand grains in a vast desert are inert objects, and due to the results of aerodynamics and various physical forces around the particles (such as friction), they form a rippled dune. Subunits can be organic or non-organic entities. In the case of the Belousov-Zhabotinski reaction, chemical reactants are the subunits forming spiral patterns. Organic cells in slime mold forming structured tissues are active self-driven subunits. They can also be transported by other elements of the systems, for example, by the social insects. A definition of subunits within one system is not always clear in artificial systems. Metabolists produced a prefabricated capsule as a subunit of buildings. However, regarding the many emergent formations of cities, a subunit can be a single building or a single mud-brick as an atomic unit of global structures.

2.4.2 Scale

SCALE of Subunits relative to size of entire system (Malleability, pliability of subunits)

/Subunits /Size /Duration of Construction



Historical Cities:

Masonry units /small /Long, over 1000 yrs
Indiv. Building /large



Traditional Housings:

Mud brick /small /Long, over 1000 yrs



Favelas: in Brazil, etc.

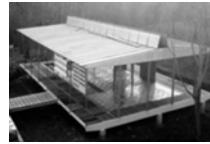
Informal (spontaneous) settlements

Various kinds of Debris /small /short ± 50 yrs
Masonry, panels, etc.



Housing in Hong Kong

Reinforced concrete/small /short less than 40 yrs



Modular systems:

Kit-of-parts /small /short
Components, joints, etc.
Fixed units



Modular systems:



Metabolism

Pre-fabricated units /large /short less than a year
Theoretically reconfigurable



Collective Constructions

Various Debris /very small /Short to Long periods



Natural Systems

/Very small /short periods
Sand Particles (Dune Formations)
Belousov-Zhabotinski reaction
(chemical reactions)

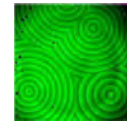


Figure 2.26 – Various scales of subunits relative to size of entire system

In order to objectively compare the size of these subunits, I propose to use a size relative to the size of the entire system as a reference for measurement. A volume of a single subunit divided by the entire size of the system is a good logical step to compare inherent characteristics of systems.

As a result, regardless of the sizes of entire systems, the natural systems tend to show smaller ratios for this relationship compared to those of many artificial systems, especially ones that were deliberately designed by professionals. Even though Bernard convection can be formulated within a small laboratory dish (a 3-inch-diameter circular vessel), the size of a fluid particle relative to an emerging size of cell cluster can be as small as a sand grain inside a dune. Termite nests are composed of varied debris transported by termites, and their sizes are much finer than replaceable capsules (room units) proposed by Metabolists. As we change the scale to urban scale, this relationship for artificial systems reduces – the size of the subunits for cities can be considerably smaller than Metabolists' buildings. When we recognize reduction in this relative size of a subunit, systems start to display more prominent characteristics of emergent behaviors. Many Metabolist and modular systems' building configurations are all predetermined by designers, but we witnessed examples of more heuristic and unpredictable growth patterns and adaptations from urban-scale artificial systems. However, examples on an urban scale require more time for growth. The graph in Figure 2.27 shows the relative size of a subunit on the x-axis and the lifetime or time for a growth period (in years) on the y-axis. Artificial systems that I reviewed occupy a diagonal zone from top-left to lower-right of the graph, and more deliberately designed systems tend to cluster at the lower right-hand side.

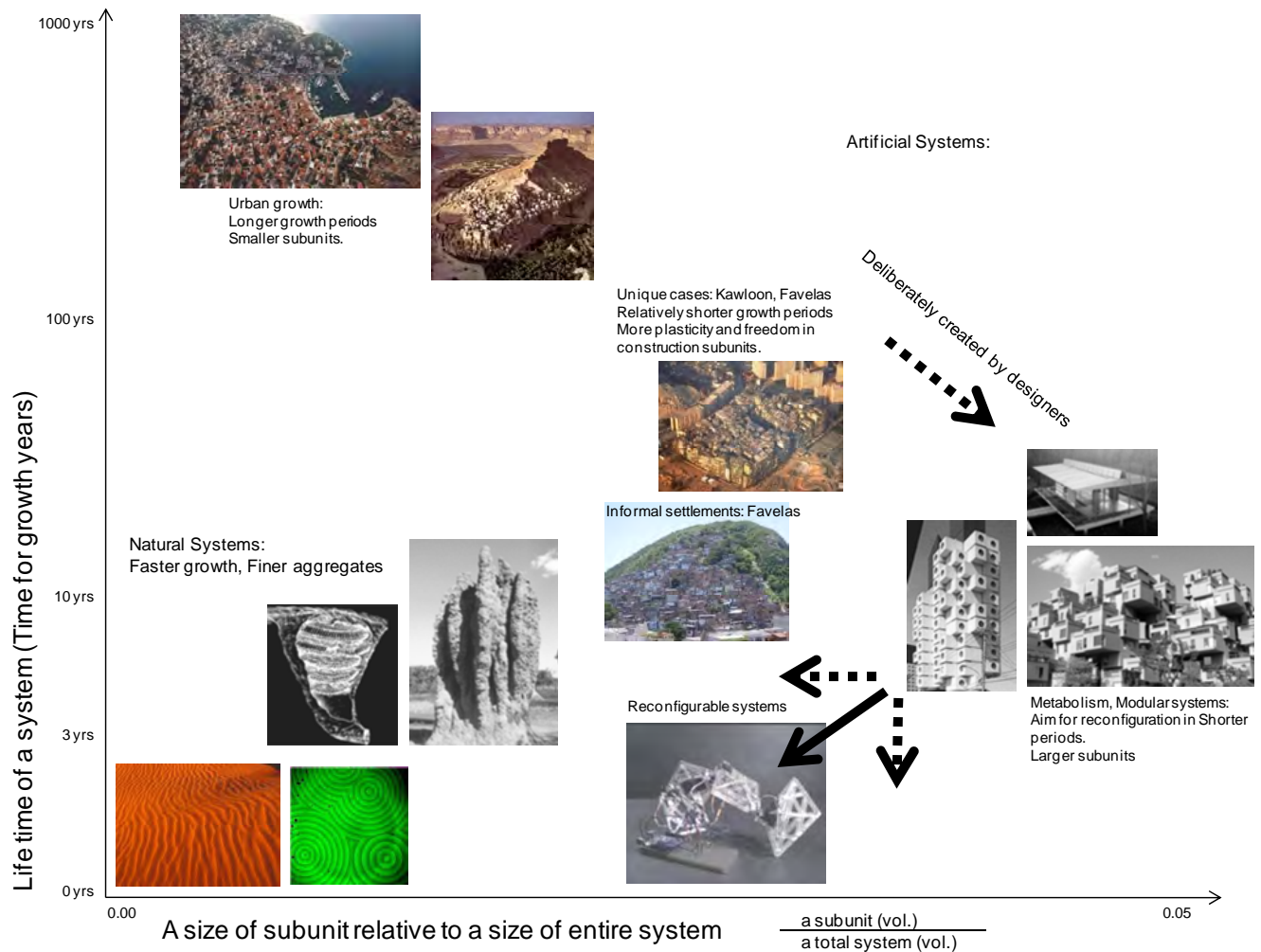


Figure 2.27 – Various Systems’ Lifetime and their subunit scales

2.4.3 Time

The lifetime of a system, or the time for a growth sequence of a system, varies depending on the system. Growth processes of cities sometimes require more than a thousand years in order to reach a fully developed stage of evolution. Growth times for natural systems vary considerably. For example, some species of trees (such as Norway Spruce) continue to live more than a thousand years, whereas micro-organisms such as slime mold grow

into an interconnected network of protoplasmic strands in a day. Artificial systems tend to require longer periods of time to actively organize themselves into a meaningful form, or to require assistance from other assembler systems. Modular or replaceable capsules are inert subunits that require assemblers such as heavy-duty cranes. Reconfigurable swarm robotics systems proposed by computer scientists are slightly more advanced systems that afford active behaviors to their subunits; thus they can be located to the left side of modular capsules proposed by Metabolists in the graph in Figure 2.27. More speculative concepts proposed by computer scientists – Smart Dusts or self-reproducing machines – are all efforts to move toward the lower left-hand corner of the graph in Figure 2.27. In order to fulfill this gap between natural and artificial systems, it may be inferred that subunits of artificial systems need to gain more active dynamic behaviors while maintaining a fine granular size relative to global systems. Recent interests in nano/micro robotics all point toward a biomimetic engineering direction, and development of such nano devices can be one way to assimilate emergence.

2.4.4 Discussion and Critique

Many formal and structural engineering aspects of natural and physical systems have been well investigated and successfully applied to various synthetic objects in our daily lives, including architecture. Natural distributions of structural forces seen in branches of trees, soap bubbles, or formations of sea-shells have been an inspiration for many architects. Structural elements in Gothic churches often mimic trees, and Gaudi in the 19th century further developed biomorphic forms to take advantage of natural force flows

in organic shapes. The formal aspect of architecture has been strongly influenced by various biological structures.

However, logistics and relatively internal aspects of biological systems, such as collective behaviors seen in aforementioned collective construction, have yet to be clearly analyzed and applied to our synthetic design processes, particularly the cases where there is no blueprint or recipe for global configurations available or defined. In the area of computer science, increasing numbers of scientists are starting to pay attention to self-organizing systems, particularly those seen in the social insects, and they try to draw on these systems to devise problem-solving methods. This approach focuses on the characteristics of self-organization, such as distributedness, flexibility, robustness, and interactions among relatively simple agents. Most of the applications so far have been in the areas of combinatorial optimization, communications networks, and control algorithms for robotics (Deneubourg et al., 1990). But even in computer science, not all of the applications are anywhere close to practicality yet, though researchers are definitely well aware of the potential advantages of the bio-inspired distributed systems to remedy many weaknesses in our existing centralized control systems. Eventually, their goal is to design artificial distributed systems that self-organize to solve problems and replace conventional centralized control by using inspirations from social insect behavior.

In architecture, few structures have ever been built or conceived based on the active application of the aforementioned logics from natural systems. Excluding some of the emergent formations of cities on larger scales over longer spans of time, adaptation of self-organization to architectural creations is an uncultivated area of study worthy of

investigation. Although there are some studies of urban-scale phenomena using agent-based models, I believe that we have yet to fully understand the latent potential of self-organization, distributed systems, and collective intelligence in the context of architectural applications. It is not simply an investigation of novelty in style, trend, mode, or superficial formal representation. It is a search for an evolution – fundamentally, a new attitude toward the creation of artifacts among us.

Ironically enough, the fact that we humans can occasionally build highly sophisticated, yet dysfunctional objects, can be considered proof of our being superior to the other species in some respects, or at least an indication of unique characteristics inherent in human intelligence. Motivations toward collective goals beyond merely functional or practical outcomes are rarely seen in any other biological species besides humans, but it is quite easy to find such examples in the field of architecture when we consider the current formalistically articulated trends in the discipline. They are an indication of our aptitudes for more advanced intellectual activities and productions. These behaviors or creative tendencies are perhaps one natural consequence of humans' being capable of storing more information; thus immense information for globally more complex configurations can be handled by individual members as a blueprint.

Nevertheless, our construction processes – even at practical levels of applications – do not seem to have reached the stage of perfection yet; they may well be on the brink of a necessary transition from conventional centralized schemes to more distributed systems. Having complete knowledge about the final global objectives in construction is getting increasingly difficult as the scale and complexity of the buildings start to exceed our

capacities for individual comprehension. Indeed, humans have limited processing speeds and cache spaces as individuals. Considering that the termites' and wasps' nest constructions require more than the period of their individual lifespans, and some of their nest sizes (in relation to body size) are way beyond the scale of any human structures ever built, we have yet to rival the magnitude of their construction scales and durations. One species of termites called *Macrotermes bellicosus* are known to be able to build structures over 30m in diameter and 6m in height (Grasse, 1984), and the size of the structures in proportion to human height is over a mile (Howse, 1970). These facts may indicate the existence of methodologies that can possibly enhance our conventional understandings of habitats.

2.4.5 Conclusions

The computational experiment by Theraulaz and Bonabeau (1995a, 1995b) introduced in an earlier section suggests that structures directed toward certain properties can be describable purely by locally implemented processes – ‘rules’ – instead of providing a complete set of blueprints, so that the structures can in principle continue to grow with the dynamic changes from the external environment or circumstances associated with the structure.

One remarkable finding from the observations of collective construction by insects is that their processes do not seem to have a discrete ‘design phase’ before the construction. ‘Design,’ ‘construction,’ and ‘operation’ are seamless concurrent activities in their processes, and these characteristics help them to gain significant flexibility in their

habitat designs. Development of flexible and adaptable architecture has been a perennial theme among practitioners, and some of the failures among the Metabolists in the 1960s (Frampton, 1992) clearly indicate the difficulties of designing universal subunits that could endlessly tolerate technological, environmental, and circumstantial changes associated with structures. In order for structures to self-assemble and dynamically grow, the scale of the movable or replaceable components needs to be examined, as our structures at architectural scales are under different magnitudes of influence from physical forces such as gravity, compared to wasp nests or cells in slime mold. Simple adaptation of distributed logics in morphological aspects of design alone will not be adequate to realize truly adaptable intelligent structures. Consistent adaptation of distributed concepts throughout all phases of projects may be an essential principle for the evolution of robust and flexible structures. Processes of coming future structures may not possess discrete phases such as design, construction, operation, reuse, and demolition; the alternative task for us may be encoding the spontaneous growth of structures as a genotype, rather than providing complete sets of static blueprints.

Chapter 3

Computational Method of Self-organization

Introduction

The primary objective of this chapter is to introduce three different types (or phases) of computational methods that are related to the concept of emergence. These methods are categorized into different types based on different objectives – evaluation, design search, and growth and adaptation.

This thesis's primary objective is in the third category of computational method: growth and adaptation. As one of the promising computational approaches for implementing growth and adaptation, the concept of self-organizing computation is introduced. In chapter 2, existing examples of self-organization were reviewed in relation to their subunits and time for growth periods; in this chapter, computational implementation and applications of these dynamic mechanisms are presented through more precise definitions of the principles and two application examples that follow.

3.1 Computational Methods: Three Phases of Computational Applications in Design

Computational Design: *Stages (phases) of applications (a system requires evaluations)*

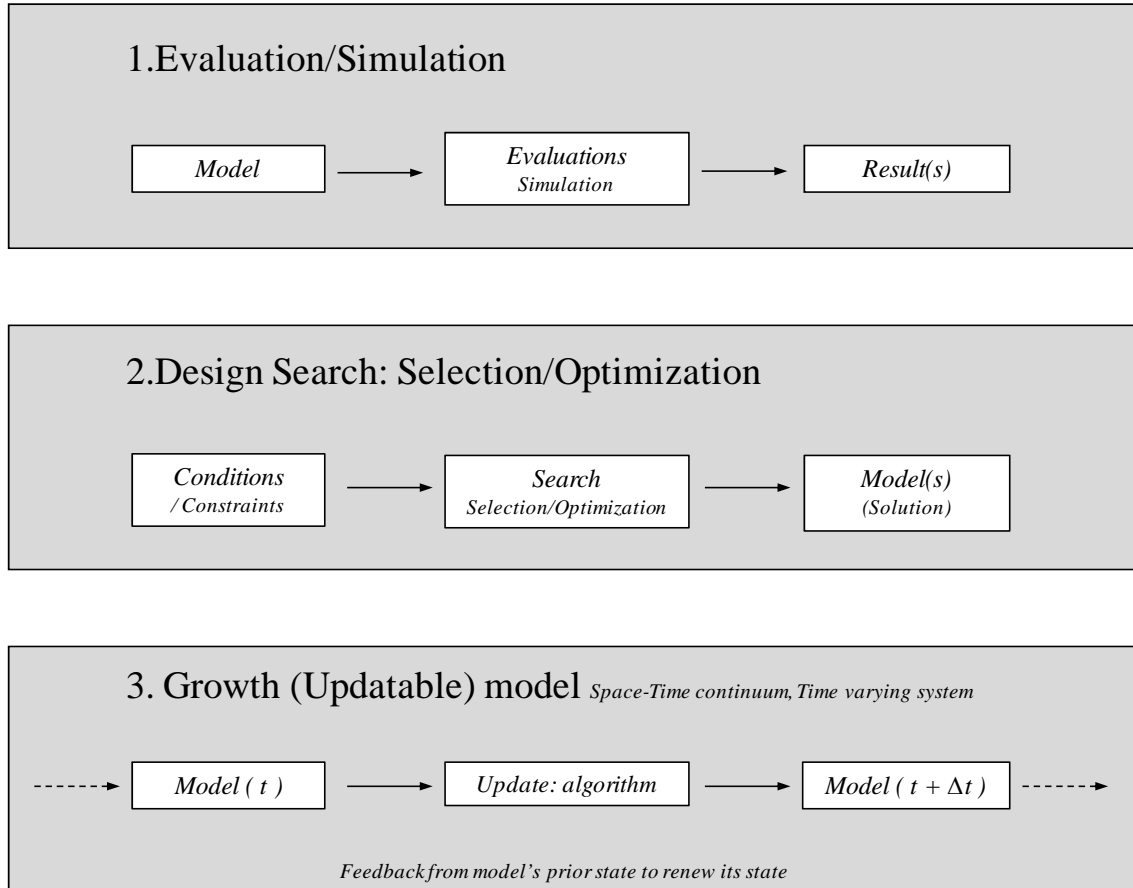
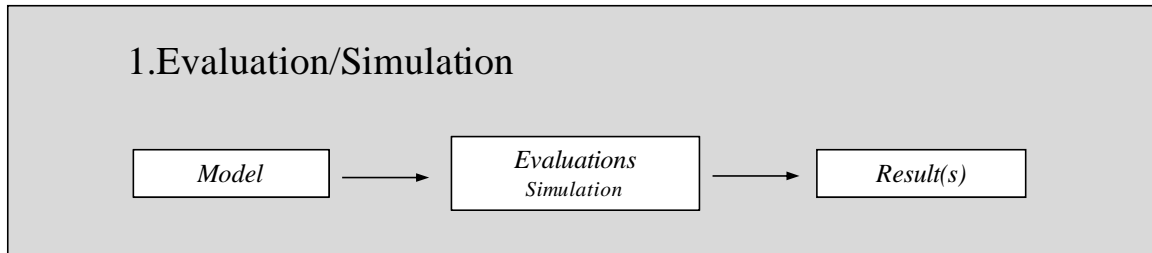


Figure 3.1 – Three phases of computational applications in design

One of the objectives of the thesis is to investigate and explore computational methods related to the concept of emergence. In order to clarify the relationship between computation and design, I categorize computational methods in the context of design into three different types based on different uses – evaluation, search, and growth and adaptation. They are not strictly separable, and are to some extent interrelated. For instance, implementation of the design search method often requires implementation of

the evaluation method in advance. Relevant existing algorithmic approaches for each category of computational methods are introduced.

3.1.1 Method 1: Evaluation



The first type of method is to use computation as an evaluation tool for a model. In this case, a model is given as an input, and the computational tool is used to analyze the model's properties, performance, and relationships to its environment. Structural analysis software is one example of this category, since it can derive a model's structural performance from a given model as an input. Evacuation simulation software is another computational application which falls into this category. Occupants' behaviors can be computationally derived from one instance of spatial configuration.

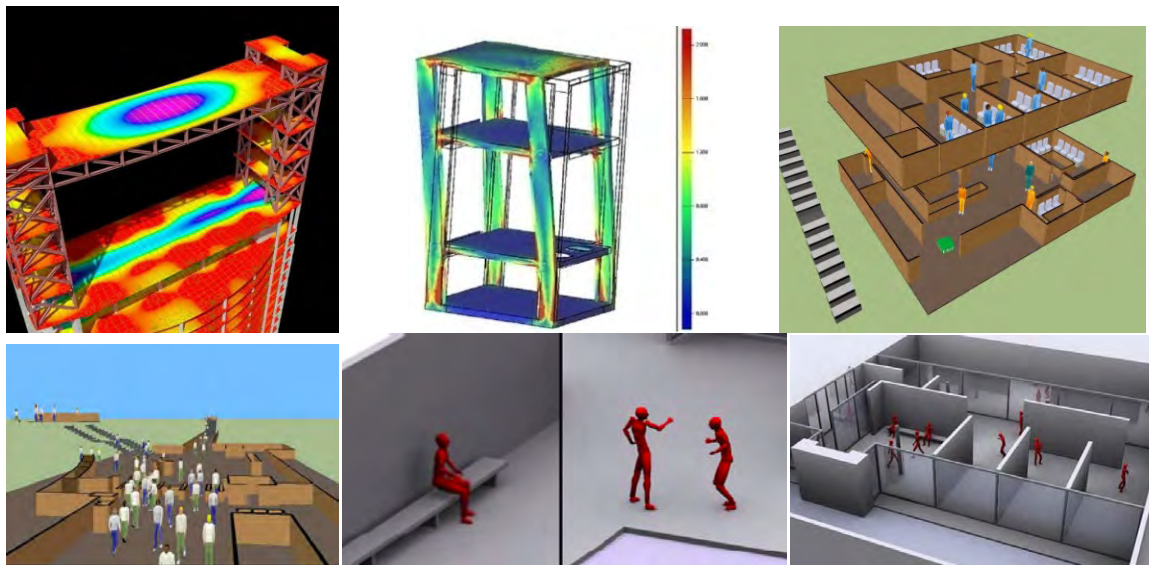
Quantitative / Qualitative Evaluation

One problem of computation in architectural applications is that the evaluation of architectural models rarely forms deterministic and analytical utility functions. Evaluation of design cannot be reduced to equations.

In the case of structural analysis software, the input model will typically deliver a single result though there can be multiple solutions through numerical computation such as

finite element analysis (FEM), and hence the result can be obtained quantitatively. When we can get this one-to-one mapping from a model to an evaluation by formulating numerical equations, evaluation is deterministic. Such quantitative evaluations are relatively straightforward.

The aforementioned evacuation simulation might provide multiple interpretations from a single spatial model, or a single run of a simulation might not be sufficient to determine spatial characteristics of a model. Escaping occupants' behaviors are often stochastically applied, based on statistical information about human behaviors under certain circumstances; every run of the simulation run can be slightly different. Normally, analyses based on stochastic simulations require millions of trials to verify their results. In this case, evaluation has, to some degree, qualitative characteristics.



Figures 3.2 – Various Evaluation Tools in Architecture:

Quantitative Evaluations: Top-left: FEM-based structural analysis tools (From www.autodesk.com), Top-middle: (From www.hitachi-kokusai.co.jp/goyo/html/kaiseiki.html) Top-right & bottom-left: Evacuation simulations software package by Forum8 co. ltd. (<http://forum8.co.jp>)

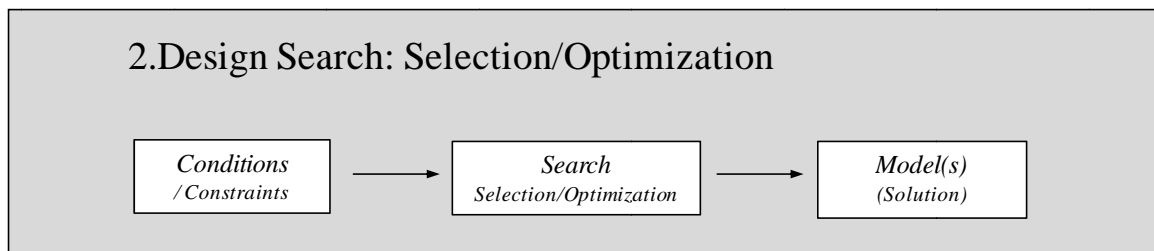
Qualitative Evaluation: Agent-based visualization tool by the author. (Narahara, 2007a; 2007b)

However, there are cases where we cannot find solutions deterministically due to the complexity or non-linearity of the problem framework. Most architectural design cannot be evaluated by a single equation. Multiple criteria need to be considered.

Even if you can formulate multiple equations for various criteria, comparisons of evaluations from multiple results are not straightforward. Sometimes, simple summation of all objectives does not represent characteristics of the particular solution relative to the other solutions. Concepts such as Pareto-optimality are often used for multi-variable evaluations. I will cover the background of this area in more detail in the next section.

Major objective evaluation criteria within architectural design evaluations – structural, environmental, regulation-related issues, and so on – can be analyzed and predicted to some extent by using today’s advanced computational tools. Nevertheless, some criteria cannot even be analytically formulated by numerical equations. Aesthetic evaluations are common issues in design practice that are in many situations computationally non-describable.

3.1.2 Method 2: Design Search: Generation + Selection → Solution



The second category of computation method in this context is design search. In this category, a computation tool derives solutions from given sets of conditions.

This is basically a reverse operation of the previous method, evaluation, which derives conditions (or inherent behaviors) from a model (a form). The design search method provides a model (a form) from certain sets of given conditions. This operation flips the arrow for the direction of derivation.

In order to computationally implement this method, we typically perform two steps. The first step is autonomous generation of design instances, and I call this operator a ‘Design Engine.’ The second step is ‘Selection’ from the instances produced. The system needs to select the solutions that best satisfy the original input conditions. This selection process requires its own evaluation methods.

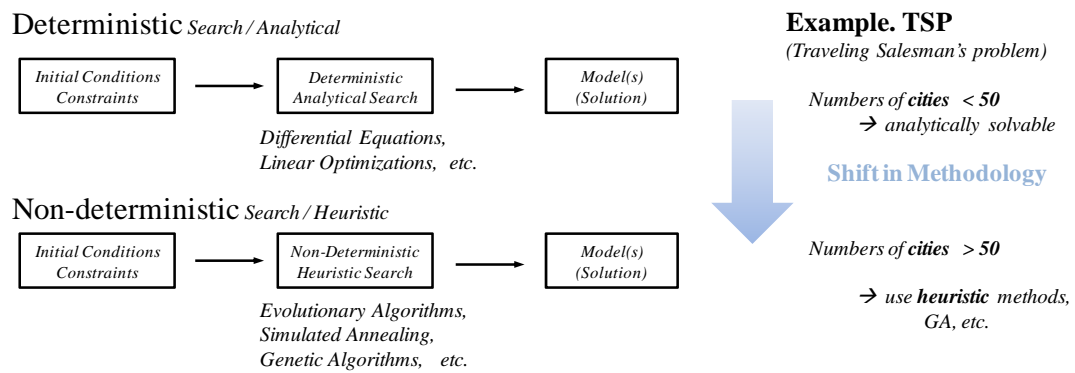
Since there is no model to evaluate or select at the beginning of the process, the computational system needs to autonomously produce or find appropriate models to consider. At this stage, instances provided by the design engine do not necessarily satisfy input conditions. These models can be all possible permutations of models that loosely satisfy a framework of the initial conditions. In some more trivial cases, evaluation methods themselves are analytically well-formulated so that they can search for solutions deterministically from all possible conditions. In the case of finding a single point in two-dimensional space, all possible combinations of two real numbers, representing a Cartesian coordinate system in two-dimensional space, can simply be a search space, or it can be certain bounded areas in a Cartesian coordinate system.

The above two categories of method – evaluation and search – have a correlation between them. In the second category, search, in order to select proper solutions, it is necessary to

be able to evaluate different sets of instances. This means that the existence of the first category, evaluation, is a necessary condition for the second category, generation, to be able to find solutions. This implies that the second category is built on the validity of the first, and that the second is a more advanced phase of computational application.

1) Deterministic / Non-deterministic Search

If the search can be reduced to an analytically solvable form such as differential equations or a linear optimization framework, it can be called deterministic. But many problems in design searches are non-deterministic: no deterministically solvable equations can be formulated from initial sets of conditions or constraints. In other cases, even if we manage to formulate analytically solvable equations for problems, the search space is simply too large to find the solutions within a reasonable time. In these cases, we have to rely on heuristic search. Auto-generation of design instances and evaluations of instances can be iterated generation after generation until the initial requirements are satisfied. Most of the heuristic methods carry out a search internally based on trial and error. Common computational methods of heuristic searches are the family of evolutionary algorithms such as simulated annealing, a genetic algorithm, and so on.

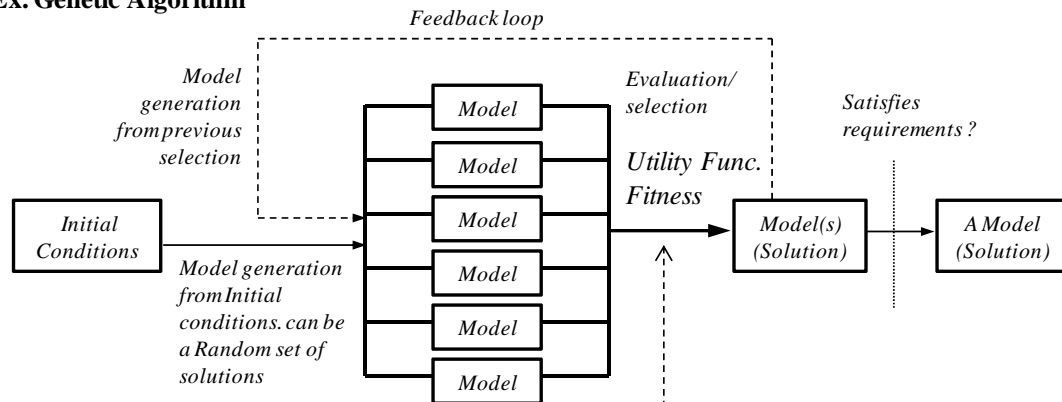


Figures 3.3 – Deterministic and Non-deterministic Search: Shift in Methodology

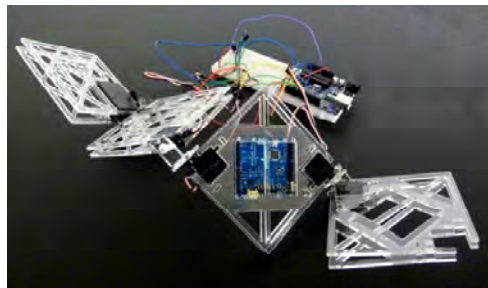
2) Genetic Algorithm (GA)

A genetic algorithm (GA) is a search technique for optimizing or problem-solving. Its mechanism is based on evolutionary biology, using terms and processes such as genomes, chromosomes, cross-over, mutation, or selection. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and those individuals are modified (mutated or recombined) to form a new population, which becomes current in the next iteration of the algorithm.

Ex. Genetic Algorithm



Ex. When you cannot use implicit objective functions.



Ex. Interactive GA

uses human interactions for selection process

Ex. Physical Testing/Measure

If conditions are too complex to simulate, it tests performance in real physical environments.

Figure 3.4 – Algorithmic flow chart of GA

Evaluation criteria are applied as fitness to select the better species to be used as a next generation. In this method, proper evaluation of feedbacks from the previous generations constantly improves the current generation.

An algorithmic workflow of GA's represents the concept of this second category, search, very well. Crossover and mutation functions serve as an instance generator – design engine – and a fitness function serves as an evaluation criterion – performing selection.

```
Initialize the population;  
Evaluate all members of the population;  
While not done{  
    Select individual(s) in the population to be parent(s);  
    Create new individuals by applying the variation operator  
    (crossover and mutation) to the parents;  
    Evaluate the new individuals;  
}
```

The standard algorithmic flow of a (canonical) evolutionary algorithm

3) Traveling Salesman's Problem (TSP)

Many problems in our day-to-day scenarios do not have a deterministic method for reaching a clear-cut solution. One of the most famous problems in this category is called the Traveling Salesman's Problem (TSP): given a number of cities on a map, to find the shortest path to visit every city without skipping or duplicating the cities to be visited. This is a good example representing a methodological shift between deterministic and non-deterministic search, based on different initial conditions of the search.

In TSP, when the number of cities is less than fifty, it is known that the problem is reasonably analytically solvable by formulating an algorithm based on linear programming. However, when the number of cities to visit exceeds fifty, it is known that

very few formulas or analytical methods are effectively available, and the “trial-and-error” type of heuristic algorithm is a computationally faster way to search for the optimum solution. Although the solutions from heuristic algorithms are only an approximation, they will provide solutions close to the optimal with high probabilities. Relying on the trial-and-error type of approach with the help of computational power guarantees practical solutions within a reasonable time. The TSP belongs to the class of NP-hard problems in the theory of computational complexity, and the aforementioned genetic algorithm is often used as a heuristic optimization technique to solve the TSP.

The above TSP is a typical example that displays the shift (or transition) of choice of problem-solving methods based on conditions or frameworks of the design problems. In the case of TSP, there is a threshold value for the number of cities that changes the effective approach for problem-solving.

I argue that the same discussion can be applied to architectural design problems. Beyond certain quantities of information for initial conditions or constraints, architectural design problems become “complex” problems. This fact implies that we need to consider the use of heuristic methods for computationally solving many problems.

In architecture, one of the main challenges today is the quantity of information and the level of complexity involved in most building projects. For example, housing projects for thousands of people have been emerging in urban areas. Furthermore, newly emerging usage and social demands for buildings have started increasing the complexity and

quantity of building programs, and additionally new technologies allow architects to conceive unprecedented spatial organizations and hierarchies of building components.

In this context, a design method led by a single intelligence or designer may no longer be reliable enough to solve complex programmatic requirements. With multiple environmental, functional, and economic constraints, it is extremely difficult for designers to find an optimal design solution out of millions of design possibilities by any existing conventional design methods. Use of heuristic methods such as genetic algorithms is one possible solution source.

4) Multi-objective Optimization Problems (MOP) and Fitness Measures

As I mentioned earlier, GA's framework requires a utility function to evaluate the population of various genomes generated by crossover and mutation processes from prior generations. This utility (fitness) function can be made up of multiple functions. In architectural selection processes, most of the cases we have to deal with involve multiple criteria – environmental issues such as a daylight factor, structural performance, density, economy, particular geometries a designer intended to achieve, and so on. These criteria form multiple objective functions, and evaluations derived from each objective are often in conflict with each other: improvement in one objective may cause detrimental effects on other criteria. Finding the best solution that will simultaneously satisfy all the criteria is often very difficult, and we may have to find compromise solutions from trade-offs among evaluations based on multiple criteria. As background, I will review three major approaches for multi-objective optimization problems (MOP).

- Plain Aggregation Methods (weighted-sum)

The simplest way to integrate multiple objectives is to combine objective functions by creating a weighted sum. This combination forms a linear function of the objectives using weighting factors based on some knowledge of the problem, and produces a single measure of merit that represents the overall performance of a solution. In this method, the weights need to be predefined by doing optimization, and the definitions of weights require some interpretations of the problems or preference for certain given information. This approach is heavily dependent on the weighting factors, and slight changes in weighting factors can lead to drastically different solutions. In architectural projects, utilities from many different criteria can be expressed by functions of cost – especially large-scale projects by developers – so that the simple linear summation of all criteria can be used as a fitness function. It is also known that the result from a single measure of merit is difficult to interpret since all objectives are mixed together in one figure. This approach may favor solutions with extreme performance on at least one objective (dominated solutions) and might not be able to find compromise solutions.

$$Fitness = (weight1) * Objective1 + (weight2) * Objective2 + ... + (weightN) * ObjectiveN$$

- Population-based Non-Pareto Approaches

According to Fonseca (1995), this class of methods seeks multiple non-dominated solutions without directly using Pareto fitness (which will be explained in the next section). The Vector-Evaluated Genetic Algorithm (VEGA) by Schaffer (1995) is one example in this category. In the first step, VEGA selects subpopulations of the next generation from each individual's objective functions. In the second step, subpopulations

favoring individual fitness from step one are combined and are used as a parent for crossover and mutation in the same way as for a single-objective GA. Then, the new population is evaluated by a linear function of the objectives where the weights depend on the distribution of the population for each generation. VEGA was one of the first attempts to carry out multi-objective optimization using GA. One shortcoming of this method is that the population tends to converge to solutions which are superior for one objective, but poor for others (Konak et al., 2006).

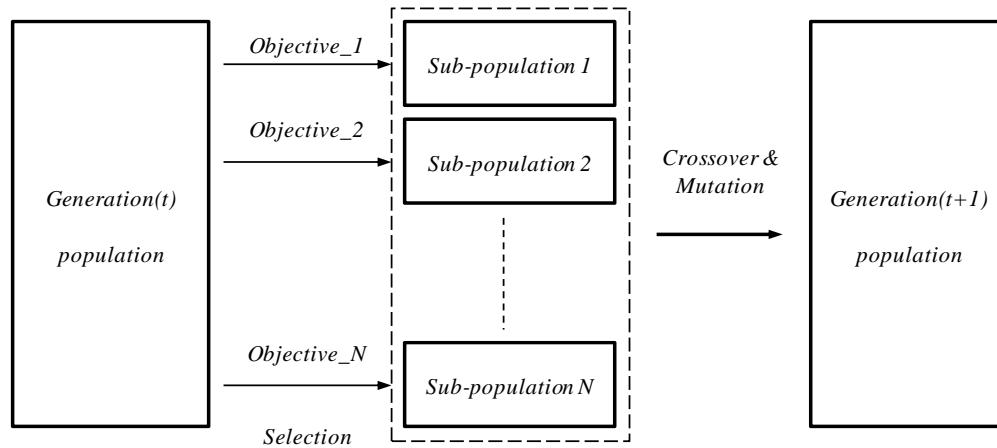


Figure 3.5 – VEGA Diagram

- Pareto-based Approaches

Pareto-optimality is a concept in economics named after the Italian economist Vilfredo Pareto (1848–1923). Pareto-optimality exploits the concept of dominated and non-dominated solutions. A solution is Pareto-optimal if it is not dominated by any other solutions. Implementation of this concept into multi-objective optimization is done as follows. A solution dominates another if it is better than the others for at least one objective function and at least as good on all the others. Using Pareto-optimality as a

fitness measure of an individual in the population, a ranking is created that can be used for selection.

The first two methods – plain aggregation and non-Pareto – are using an absolute measure of fitness from a single measure of merit, whereas fitness based on Pareto-optimality can rank the relationships among other feasible alternatives sampled by GA. This way, the algorithm can find compromise solutions more readily, and typically Pareto-based approaches do not require any prior knowledge about preference for decision-making process. (Weights for the simple aggregation method represent preferences in the decision-making process (Caldas, 2001; Konak et al., 2006).

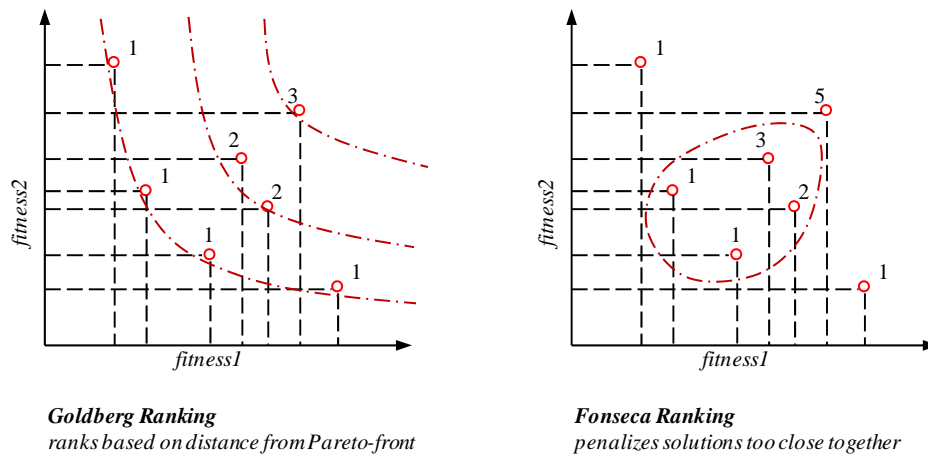


Figure 3.6 – Pareto Ranking methods by Goldberg and Fonseca

5) Qualitative Evaluations

There are many cases where one cannot form implicit objective functions for given problems. Especially in architecture, many evaluations are based not only on quantitative but also qualitative evaluations. In such cases, there are several strategies to compensate for the lack of implicit objective functions.

- Interactive GA

Use of human evaluation as a fitness evaluation in GA's framework is one strategy when mathematical fitness functions cannot be formulated for certain utilities. Examples of this application include evaluations of various artistic designs and forms to fit a user's aesthetic preferences. Evaluations of many qualitative criteria involve human perception. For instance, when we consider selection of specific geometry, characteristics such as symmetry can be straightforwardly implemented by mathematical expression, but selections based on aesthetic judgments do not always have efficient mathematical expressions to describe them. For some cases, replacing mathematical equations with human evaluation is more efficient and effective. In this method, GA's program has an interface that allows users to choose schemes during the course of the evolutionary run. The chosen schemes' characteristics will be inherited as genetic features for the next generation. This selection by humans can be done intuitively without any mathematically explicit descriptions. This process allows having a dialogue between a human and a computer system to co-evolve designs. Solutions for complex problems derived by heuristic methods are "better" solutions, but choosing a quantitatively slightly better solution does not always guarantee the most appealing solution for human perceptions.

In architecture, building schemes that fit the pragmatic and quantitative requirements, such as zoning, values for lighting, or adjacency conditions, result in different topological and geometrical variations. Impacts of subtle differences in fitness scores often become meaningless, when resulting variations in configurations are dramatically different. In

such cases, adaptation of feedback based on human perceptions is more efficient for finalizing a direction toward specific convergence of design.

- Feedback from Physical Experiments

Instead of simulating and computationally calculating fitness, getting fitness from an actual physical model is a quite effective method for some special applications. In the robotics area, searching for a proper walking motion for a robot can be acquired from an actual robot's movement as a fitness value. Simulating a robot's movements completely inside the computational environment is often very difficult. Simulation of all physical dynamics in a real-life environment is extremely cumbersome, and it will never be the real environment. Direct feedback from real life is the real information we need, and sometimes this is the faster path. Bongard, Zykov, and Lipson (2006) demonstrated this logic from their robotics research, but there has been little work done in this area in architecture. Since evaluations of architectural instances can be highly influenced by many physical environmental factors, such as lighting, wind forces, heat flows, and so on, this approach has great potential. In particular, performance-based evaluation of kinetic architectural components, or operations of buildings related to environmental criteria such as day lighting and shading can be promising areas of application. I will introduce my implementation of this logic later in the thesis.

6) Projecting an Arrow from Method 1 to Method 2

I mentioned the correlation between 'Evaluation' and 'Search' methods and their reversible characteristics. I would like to elaborate this notion in the section below.

- Bayes' Theorem

One practical example of the concept of arrow flipping can be seen in computational applications using Bayes' theorem. Bayes' theorem shows the relation between one conditional probability and its inverse; the probability of a hypothesis, given observed evidence, and the probability of that evidence, given the hypothesis. In other words, Bayes' theorem can mathematically represent the relationship between the conditional probability of event A given B and the converse conditional probability of B given A. For example, the probability of a certain diagnosis given a certain probability of symptom (posterior) can be derived from the probability of the symptom given the diagnosis, the probability of the diagnosis (prior), and the probability of the symptom (marginal probability). In many real-life scenarios, there are cases where finding $P(B|A)$ is much easier than finding $P(A|B)$, and this technique is widely used for inference of many events represented by $P(A|B)$. Many spam filters are designed based on this concept, and many search engines, such as Google, also use it in order to infer information that users are looking for from knowledge based on data from users. This is a practically successful computational application that flips the arrow of derivation.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (P(B) > 0) \quad \dots \text{Bayes' Theorem}$$

- From Pedestrian Simulation to Topology Optimization

Recalling the section on lane formation and pedestrian simulations at the very beginning of the thesis, I would like to explain the concept of arrow flipping using an example from that area.

A surprising result concerning pedestrian simulation can be found in evacuation egress studies using cellular automata by Helbing (2000) and Kirchner, Nishinari, and Schadschneider (2002). Their simulation models showed that placing a column in front of the exit slightly shifted from the center (to the left or right) can improve evacuation times considerably. Under certain conditions, the column does not act as an obstacle. Kirchner et al. explained that the column subdivides the pedestrian flow and reduces conflicting situations close to the exit. Normally, a column has a screening effect that forces pedestrians to take a detour and hence potentially increases evacuation times. Kirchner et al. demonstrated that there is a non-trivial dependence of the evacuation time on the column location.

The above experiments by Helbing and Kirchner et al. showed that morphological characteristics of architectural space can affect occupants' behaviors. They can computationally evaluate and predict these behaviors to some extent from the schematic layout of the room using their simulation tool. This fact implies that, in principle, we can derive optimally preferable spatial configurations by providing desired behaviors as input information. If we can somehow iterate through potential spatial configurations, and evaluate each one of them using a simulation tool, we can reverse-engineer the process and flip the direction of the derivation arrow in the figure below. Normally, the search space of solutions is too large, so that there is no way we can use evaluation or simulation to get all the possible answers. This flipping of the arrow concept has been used in some engineering applications as a topology optimization technique based on structural forces or aerodynamics. For example, two software applications by FE-Design GmbH (2010) in Germany, TOSCA and ABAQUS, are Finite Element Method (FEM)-based structural

optimization software that are used to optimize Formula One racing automobiles' body shapes based on computational fluid (aero) dynamics.

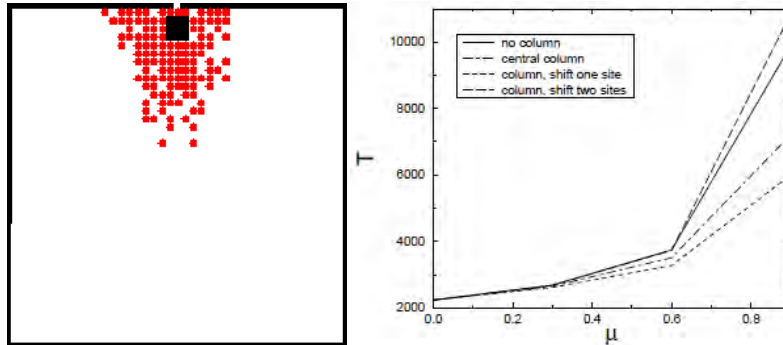


Figure 3.7 – An evacuation of a room with one door at the middle of the top wall with an additional column of size 3×3 cells placed in front of it (left). Evacuation time as function of the friction parameter μ for four room geometries: no column, central column, and column shifted by one and two lattice sites (right). (Kirchner, Nishinari, and Schadschneider, 2002).

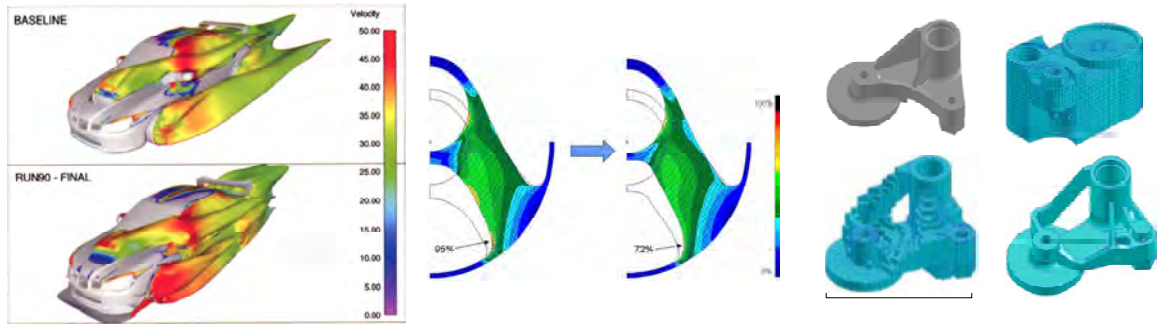


Figure 3.8 — **Examples of Topology Optimization:** Left from www.optimalsolutions.us. Middle & right from www.fe-design.de (Fe-Design GmbH, Germany). Original design inputs by users are optimized based on stresses in members.

- Flipping the Arrow

The above examples represent ‘*evaluation*’ as a necessary computational implementation in order to build a computational system that can ‘*search*’ for solutions autonomously from given sets of conditions.

Figure 3.9 shows abstract diagrams that illustrate the concept of arrow flipping. In the left diagram, the square at the middle influences the flows of particles around the square,

producing certain resulting particle trajectories (behaviors). Particles in the diagrams can be abstract representations of any type of active component: representations of aerodynamic flows, pedestrian agents, and so on. In this case, a certain geometrical or material property of the square (model) induces certain corresponding particle behaviors. Conversely, if one knows desired behavioral patterns that need to be induced by a model, it is natural to search for certain models that can produce the target behaviors. The methods introduced in this chapter, such as GAs, are some of the techniques that can potentially be used for such inductive tasks. This concept is abstractly represented in the right diagram of Figure 3.9a. Note that GAs require an ability to *evaluate* instances using a fitness function in order to *select* or *search for* certain solutions. In pedestrian simulations, if one prefers a certain pedestrian movement pattern as a result of certain models, there are possibilities to inductively search for such models, as this section has explained.

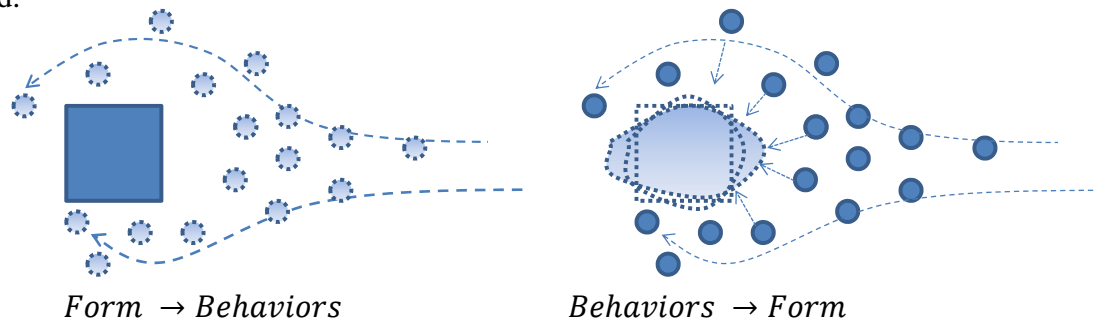


Figure 3.9a – Concept diagram of Flipping the Arrow by the author

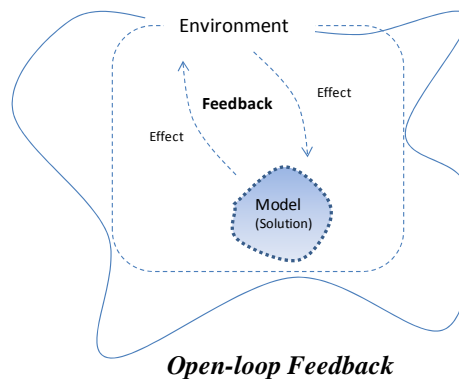
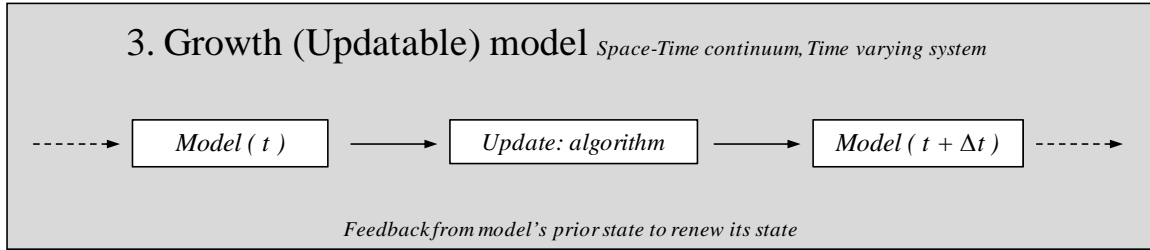


Figure 3.9b – Concept diagram of open-loop feedback

3.1.3 Method 3: Growth and Adaptation



The third category of computational method is ‘Growth and adaptation.’ This method provides a computational description of gradual growth processes over time. In this process, a model and its environment have a reciprocal relationship (See Figure 3.9b in the previous page). A model is first created by conditions and constraints inherent in its environment. Then the model’s behaviors and growth influence the environment and start to change it. This change in the environment becomes a new incentive for the model to update itself to conform to its new environment. This perpetual feedback between the model and the environment is a continuous loop in time series. This process can be implemented as a computational model by providing an algorithmic description for the model to update its state.

In principle, if we can write a general procedure for a model at arbitrary time T to renew its state at time $T+\Delta t$, this model can continue, by updating its state, to grow. This procedure for updating needs to be conditionally applied, based on the states of the environment, which implies that the description of ‘self’ is not adequate for the description of the model in this category. Such a model needs to be equipped with perceptions of environmental conditions in order to produce its next action. These sensing and action functions are the essential behavior for the model inside the spatiotemporal settings.

1) Fibonacci Sequence

The Fibonacci sequence is a simple model where each successive term of the sequence (after the first two numbers) is defined as the sum of the two preceding terms. Although this is not an architectural model, the Fibonacci sequence represents the logic of growth described above very well. Once initial conditions and a recurrence relation that recursively defines a sequence are given, the Fibonacci sequence can continue indefinitely. Of course, this type of recurrence relation has to be more complexly and conditionally applied for many models in the real-life environment. A logistic map is often used for a population growth model in biology, and is also described by a non-linear recurrence relation.

$$\begin{aligned}
 \text{Fibonacci sequence} &:= 0, 1, 1, 2, 3, 5, 8, 13, \dots F_{n-2}, F_{n-1}, F_n, \dots \\
 F_n &= 0 & (n = 0) & \text{(initial condition)} \\
 F_n &= 1 & (n = 1) & \text{(initial condition)} \\
 F_n &= F_{n-2} + F_{n-1} & (n > 1) & \text{(a recurrence relationship)}
 \end{aligned}$$

The two previously discussed methods – evaluation and search – are single-shot events without any particular relation to time and growth. These methods represent our current mentality toward design practices: search and evaluation for a particular design solution is always considered in the static context of the moment, but not in a dynamic context. This characteristic reveals modern planning methodological tendencies and some limitations as well. Active adaptation to ever-changing environments has not been a critical agenda for many building types until quite recently. Demands for such methodologies have started to increase, as the complexity and quantities of architectural programs in contemporary society increase.

In theory, without having any complete big pictures of final outcomes, systems can continue to grow and maintain globally functional states by using the spatiotemporal procedures. In this case, design is not a convergence toward any predefined goal; instead, solely spatiotemporal procedures can lead the growth in appropriate directions based on sensing from the current conditions without imposing specific pre-defined design templates.

2) Evolutionary Game Theory

I would also like to note that evolutionary game theory can be another potential computational technique for describing dynamics of growth and negotiations among various multiple building types.

In the 1970s the prominent biologists John Maynard Smith and George R. Price applied game theory to their field. Maynard Smith introduced the notion of Evolutionarily Stable Strategy (ESS). ESS uses a payoff matrix in game theory as a frequency-dependent fitness among the different species or strategies. This notion had never existed in the traditional game theory centered on the concept of Nash equilibrium. The traditional game theory was mostly based on the premise that all the players play completely rationally, whereas evolutionary game theory provided an alternative method to explain the long-run outcomes which arise from the interactions of less than fully rational agents, yet yield optimality over time.

Evolutionary game theory was the result of a unique interpretation of game theory by biologists and added a new repertoire for analyzing advantageous mutations and

evolutionary stable strategies (ESS) among groups (populations) over time. The evolutionary game theory considers ‘repeatedly played games’ and introduces the notions of mutation and selection, whereas traditional game theory is mostly concerned with single or discrete numbers of games. Individuals interact randomly with others based on payoff, which is interpreted as fitness. Successful results of games lead to greater reproduction of those strategies in proportion to others among the population. As a common method to analyze evolutionary dynamics in games, replicator equations are used to study infinite populations in continuous time. Nowak (2006) introduced various model applications of the theory, such as mutation of cancer cells and the AIDS virus, development of the grammatical structure of languages, and so on. He also introduced a spatial application integrating evolutionary game theory and cellular automata.

The spatial extension of evolutionary dynamics has been studied by Nowak and May (Nowak and May, 1992; Nowak, 2006) using a local interaction model in which each individual plays a cooperator – defector game among its neighbors inside a square lattice space. These spatial models provided a different interpretation of evolutionary dynamics from those using the replicator dynamics and shed light on potential future applications in models for more architectural purposes.

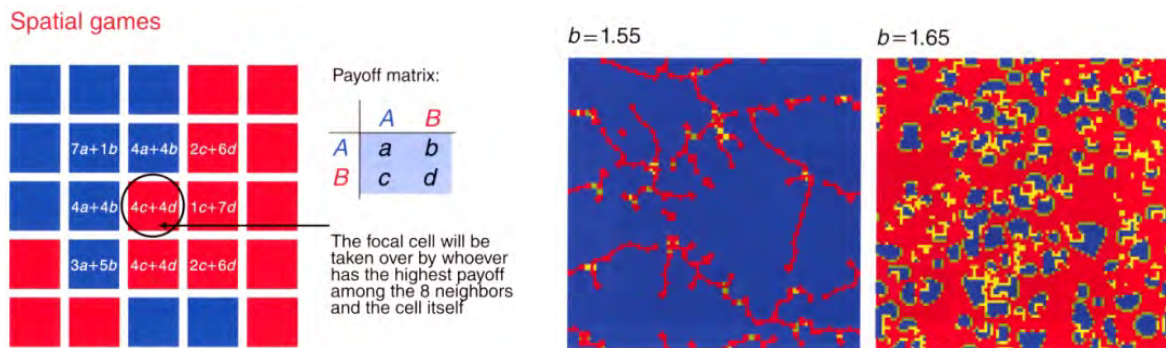


Figure 3.10 – Spatial Games (Nowak, 2006).

3) Examples:

I list the following three examples of this method in order of level of application.

A). Conventional Renovation Scenarios

Renovation and addition to existing structures is the most common and the most primitive scenario of application of this logic in architecture. In many cases, these transformations are not planned at the time of the initial construction of structures and are triggered by unanticipated changes in environments, occupancy, and population density in later periods. Normally, these alterations are executed at a certain discrete time step at once, and there is no continuity between successive transformations. Brand shows various transformations of buildings over time in his book *How Buildings Learn* (Brand, 1994). These examples belong to the aforementioned ‘closed-planning’ category (Isozaki, 1967; 1972). Discontinuity in growth patterns and the lack of bidirectional spontaneous feedback between buildings and environments characterize this class.



Figure 3.11 – The Mrs. Robert Louis Stevenson-Lloyd Osborne House in San Francisco. Example of building growth by renovations and additions. (Brand, 1994)

B). Modular System (Kit-of-Parts)

This example belongs to the ‘open-planning’ category. All the future transformations, reconfigurations, and replacements are planned at the initial stage of the construction;

however, these changes are pre-defined or constrained by the system's own physical limitations. Metabolists' buildings' infrastructures allowed some reconfiguration patterns of units, yet the growth was limited within the extent of the system's own capabilities. In general, modular systems consist of modular building blocks and infrastructural arteries that can support and combine them. At this level of application, these subunit blocks do not possess active behavior and sensing capabilities able to achieve un-programmed or unplanned configurations.

C). Self-organizing Growth

This is a more advanced application of the growth logic. The subunit is designed flexibly and universally enough so that aggregations of the subunits can adapt to many unpredictable scenarios. In order to achieve this level of flexibility in global structure, the subunits need to have some means of active mobility by having actuation devices within themselves or by relying on other devices for transportation. Collective construction by termites is one extreme example of such structures that do not require any pre-defined configurations. Procedural instructions alone can continue the construction processes. We do not know the exact logic behind them, but the models by Theraulaz and Bonabeau (1995a; 1995b) show that similar constructions can be obtained from mere locally embedded rules. Swarm robots or reconfigurable robots created by computer scientist groups also belong to this category because they have distributedly controlled subunits with sensing and actuation capabilities. In the following section, computational interpretations of these dynamic mechanisms are introduced in detail through more precise definitions of the principles and two application examples that follow.

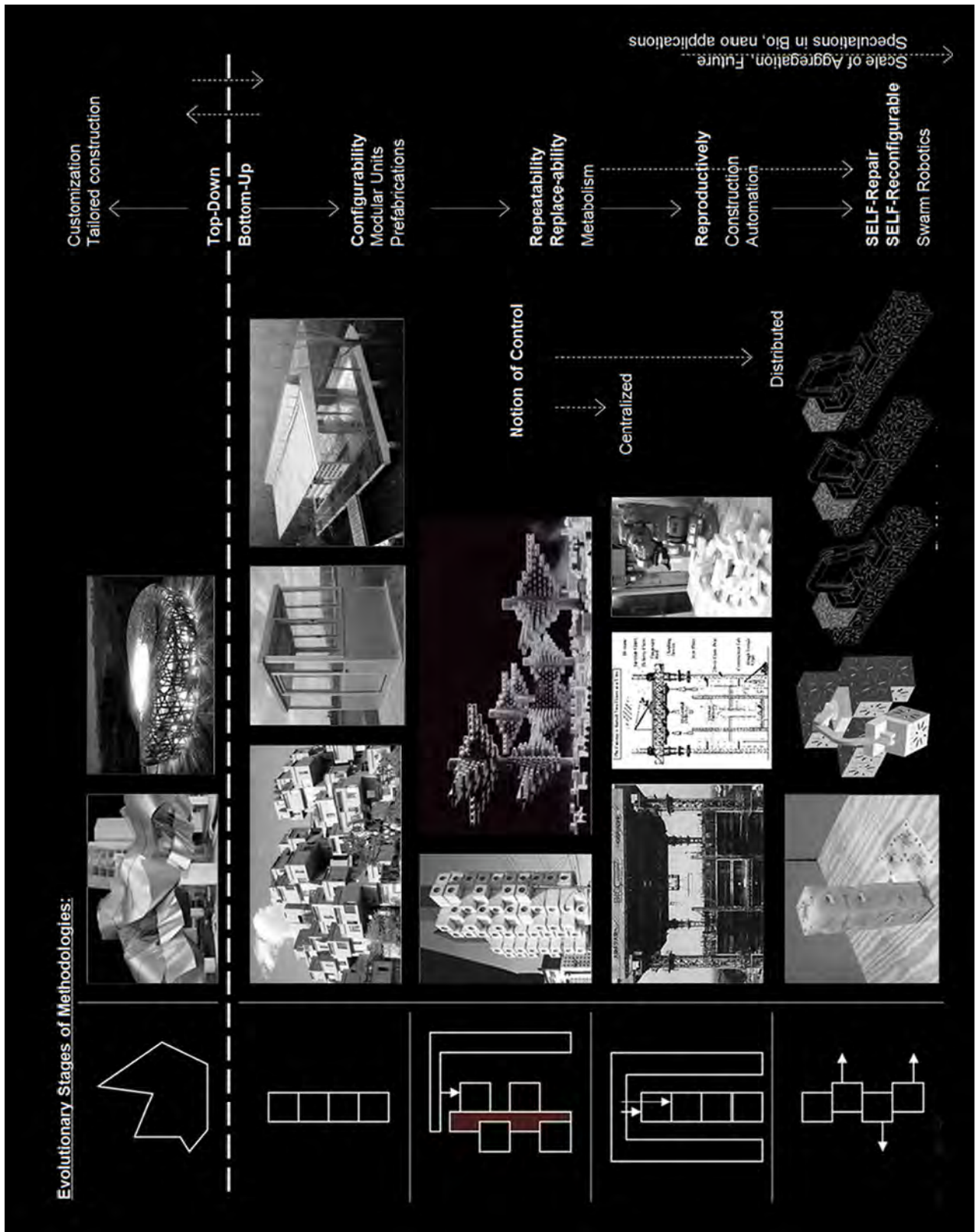


Figure 3.12 – From Modular system, to Self-reconfigurable system, to Self-organizing system

3.2 Self-organizing Computation

3.2.1 Self-organizing Computation

Self-organizing computation is a computational approach that brings out the strengths of the dynamic mechanisms of self-organizing systems: structures appear at the global level of a system from interactions among its lower-level components. In order to computationally implement the mechanisms, the system's constituent units (subunits) and the rules that define their interactions (behaviors) need to be computationally described. These interactions are executed on the basis of local information rather than being a property imposed by knowledge external to the system. The system expects emergence of global-scale spatial structures from the locally defined interactions of its own components. In pedestrian simulations, these subunits can be pedestrian agents with certain movement behaviors, and interactions of these can produce globally defined formations such as lanes, which will be reviewed in the next section.

3.2.2 Key Attributes of Self-organization

Self-organization is generally known to rely on a few basic attributes. For example, positive feedback, negative feedback, amplification of fluctuations, and multiple interactions are four ingredients listed by Bonabeau et al. (1999). Therefore, computational implementation of these attributes is essential for the development of self-organizing computation systems.

Feedback loop is basic behavior that promote emergence of structures. Feedback occurs when the response to a stimulus has a certain effect on the original stimulus. Generally, there are positive and negative feedbacks. Positive feedback means a response enhances the original stimulus, whereas negative feedback means a response diminishes the original stimulus. Together they act as an amplifier (self-reinforcer) and a regulator inside a system. These are similar to the concept of activator and inhibitor from the Turing patterns that are reviewed in the previous chapter. Stable collective patterns within a system are maintained when positive and negative feedbacks counterbalance each other.

Randomness or amplification of fluctuations is one of the essential attributes of self-organization. Randomness plays a key role in self-organizing systems, enabling them to discover new solutions. The ant foraging phenomenon illustrates this notion well. Ant foragers sometimes lose their trails by error, but this error gives them opportunities to find unexploited food sources, and to recruit others to these new food sources (Bonabeau et al., 1999). If search spaces for problems are extremely large, and if it is not feasible to check every possible solution, randomness helps systems to distribute seeds for searches and to promote growth toward unexploited directions. This implementation of fluctuations also prevents systems from stagnating at local optima. Although such a phenomenon may seem inefficient, implementation of randomness or fluctuations can be crucial where no deterministically defined problem-solving means are available.

However, randomness is not a necessary condition for emergence of structures. In some systems, without relying on any stochastic behaviors and interactions of systems' constituent units, some global structures can be deterministically gained, and these

systems can be still considered as self-organizing systems by satisfying the other attributes, such as open feedback and multiple interactions. For example, in the second example in the following section, see circle packing using a bubble meshing method. Besides initially randomly defined conditions of circles, simulated physical interactions of circles do not have any random components – yet the example’s derivation of global configurations can be interpreted as a self-organizing behavior.

Multiple interactions are another key attribute that self-organization relies on. Although ordered structures can be formed by a single individual in certain cases, self-organizational systems generally rely on possessing a sufficient density of mutually tolerant individuals. Interactions of individuals cause results of their own activities that may influence others’ activities in a system. Such correlations and interdependencies among constituent composing units of self-organizing systems are realized by the multiple interactions.

3.2.3 Key Properties of Self-organized Phenomena

Phenomena resulting from self-organizing systems usually show the following key characteristics. One characteristic is that the resulting emergent spatiotemporal structure is created from an initially homogenous medium. This means that the constituent components of a system initially possess common characteristics or similar behaviors to each other, although they may differentiate and diverge gradually during the course of interactions among them.

Another characteristic for the emergent spatiotemporal structure is the possible coexistence of several stable states. This is also called multistability (Bonabeau et al., 1999). Due to the reliance on amplification of random deviations in self-organizing systems, these deviations may cause further deviations over the course of processes and possibly produce convergences to several different stable states. Randomly produced initial conditions thus may lead the system to acquire different states.

These states can further differentiate depending on some parameters, and some will become stable states. These parameter-dependent changes in behaviors of self-organizing systems are often dramatic, and the existence of these bifurcations is one other key property of self-organization. These changes are not constant and gradual changes that can be understood as a linear summation of the individual of successive inputs. These changes often are observed abruptly when internal properties or a system's parameters reach certain threshold values, and the system starts to display qualitatively new properties. This underlying property is often referred as the “nonlinearity” of self-organizing systems.

3.2.4 Emergent Formations: Examples of Self-organizing Computation

The following section presents two relatively simple yet explicit examples of self-organizing computation in order to clarify the preceding more conceptual and abstract discussions about self-organization. Close packing of circles is one typical case where simulation using dynamics excels the performance of any analytical means, and it can be implemented by relatively straightforward codes. Simple, locally implemented physical

motions of bubbles – pushing and squeezing against each other – can eventually lead a group of bubbles to form a globally cohesive structure. Lane-formation using agent-based pedestrian simulation is another clear example that represents the concept of self-organization. Groups of pedestrians can find efficient walking formations solely from locally embedded individual behaviors without imposing any global geometry. Circle packing and lane formation are two simple examples that exhibit principles of self-organizing computation.

3.2.5 Lane Formation

“Lane formation” is a fascinating emergent phenomenon we can observe from simple agent-based pedestrian simulation. In crowds of oppositely walking pedestrians, the gradual formation of varying lanes of pedestrians moving in the same directions are observed. This is an empirically observed phenomenon and has been recorded in many real-life locations such as crowded pedestrian streets crossing in the city of Tokyo (Figure 3.13). The emergence of this spatiotemporal pattern is a result of non-linear interactions among pedestrians.

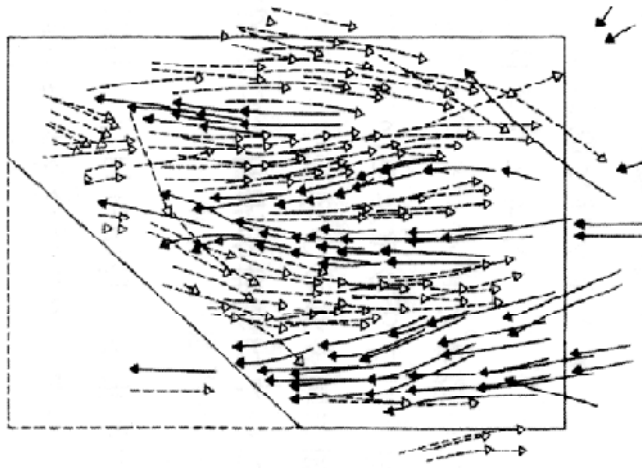


Figure 3.13 – Lane formations observed at an intersection in Tokyo, Japan (Katoh, 1980).

Each pedestrian's embedded behavior is simply avoiding others blocking his or her way in local neighborhood conditions, yet the global collective behavior that emerges from interactions of simple behaviors shows self-organized characteristics of a crowd. Emerging patterns of lanes are not externally planned, prescribed, or organized, either in computer simulations or in real-life scenarios. Interactions of only locally implemented individual behaviors gradually form globally functional and cohesive organizational structures. According to Helbing (2001), the reason for lane formation is the consequent decrease in the frequency of necessary declaration and avoidance maneuvers, which increases the efficiency of the pedestrian flow. The pedestrians are obviously minimizing the collisions among them to reach more efficient steady states by gradually forming lanes by following each other. Also according to Helbing (2001), lane formation's collective pattern of motion displays symmetry-breaking phenomena. The model has absolutely no bias for the probability of pedestrians' left turns or right turns for avoidance, and they are completely symmetrically defined. (This applies to the simulation by the author as well.)

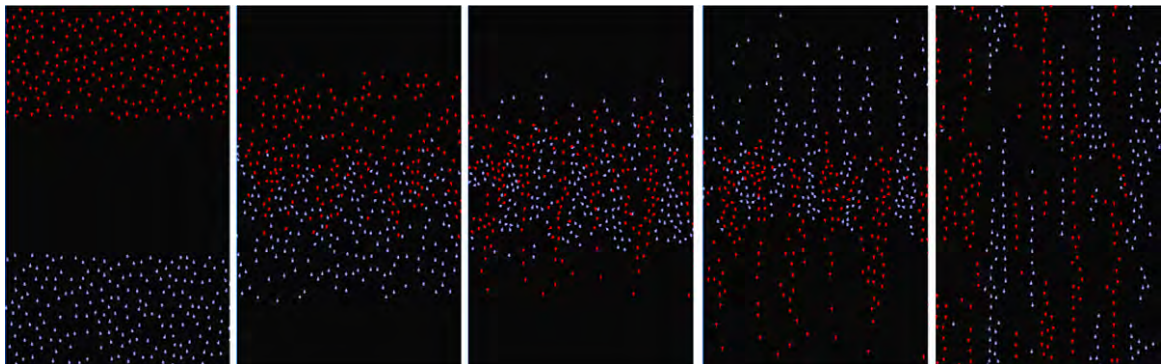


Figure 3.14 – Pedestrian simulation by the author and emergence of lanes.

The varying formations of lanes are highly influenced by the population density of pedestrians, width of the street, and field of vision implemented in each pedestrian agent. (See author's results from the experiments section below.) The basic behaviors of each pedestrian are fairly simple – they all want to avoid the people walking toward them and not to slow down their walking pace unnecessarily. The following behaviors have been observed in many past studies relating to the density conditions of real pedestrian movements (Nihon-Kenchiku-Gakkai, 1994), and the simulation by the author is based on the implementation of these behaviors. (See section below for a detailed description of the experiments.)

- Pedestrians pay attention to their immediate neighbors and avoid collisions with them. They adjust their walking speed based on the density condition around them.
- When density is low, pedestrians tend to keep a certain distance between themselves and the other pedestrians ahead of and behind them. Then they pay attention to the distances to their left and right.
- When density is high, pedestrians tend to reduce the distance ahead of them. Thus, spacing among them is reduced.
- Pedestrians survey the flow around them, and follow the direction where more people are moving in the same direction as they are heading. (Density here refers to the density in terms of the local conditions around the pedestrian, not the overall density.)

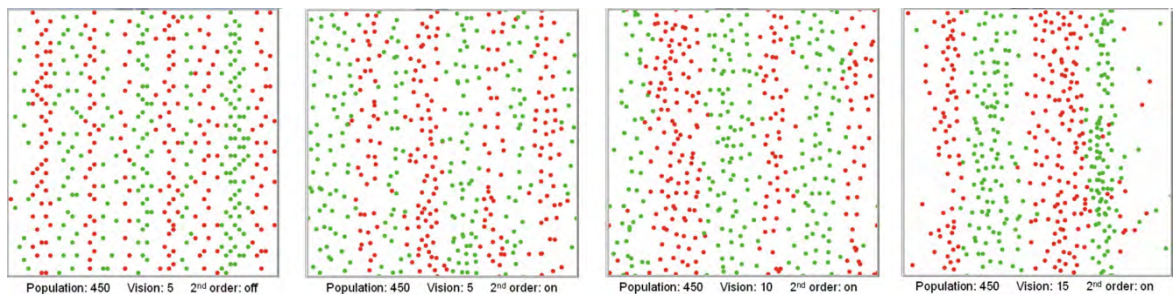


Figure 3.15 – Simulation results from various different parameter settings by the author.

The last characteristic introduces the idea of the visual ability of the modeled pedestrians. This visual ability allows the modeled human to detect the motion of the others around itself, and make proper judgments for its next movement. The extent of this vision has an inversely proportional relationship with the density conditions around the modeled human. When the person is surrounded by many people, he or she possesses a smaller field of vision, so his or her decision will be based more on local, immediately neighboring conditions. When the density is low, the person has more ability to survey the movement on a macro scale and consequently to recognize groups of people moving toward the same direction in a larger area. This behavior, and the algorithm that I have interpreted and implemented in this exercise, are similar to those of flocking behaviors.

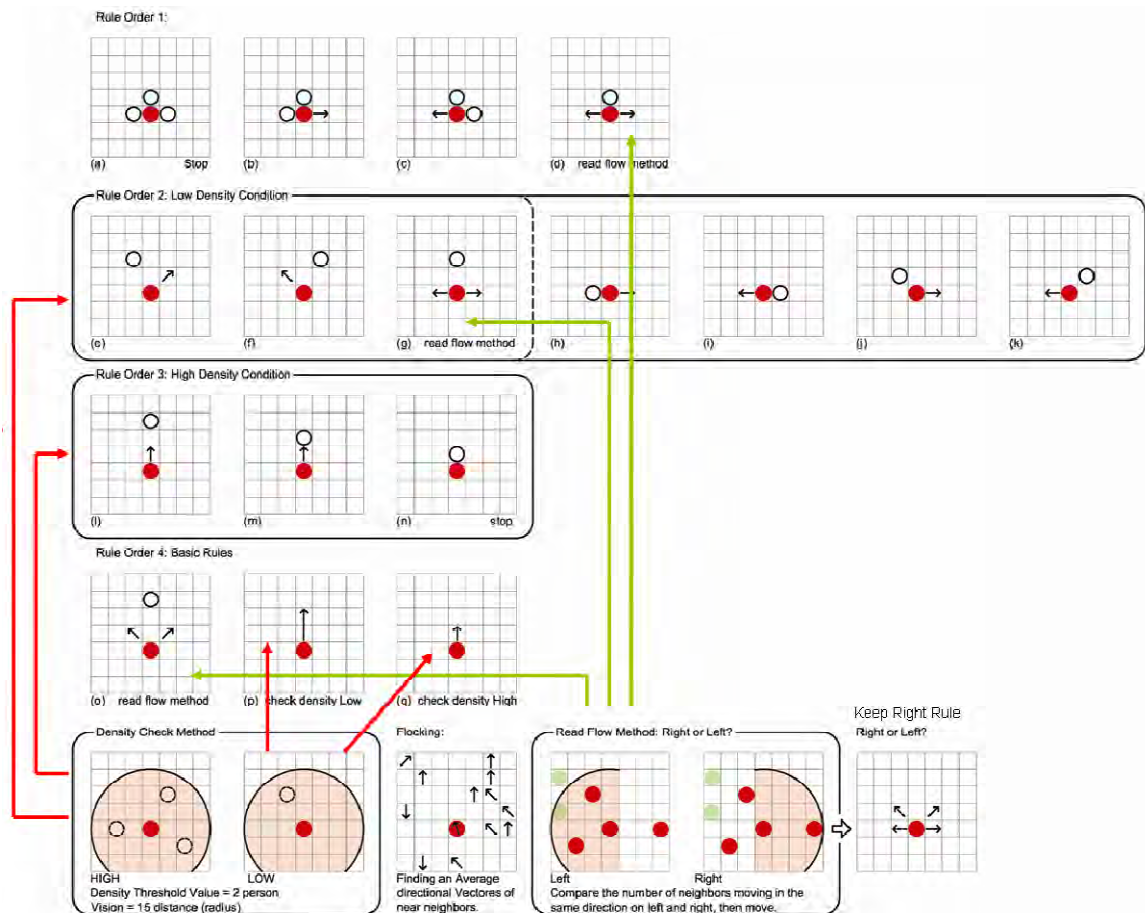


Figure 3.16 – Pedestrian movement rules for simulation program by the author.

1) Pedestrian Crossing Simulation by the Author

The following is the detailed description of my algorithmic interpretation and results of pedestrian behaviors at the crossings. My interpretation may not necessarily be alike in terms of algorithms, compared to studies by others. The aim of this exercise was to reinterpret known human behaviors under certain circumstances in the simulation model by the use of algorithms.

Based on various populations (density), conditions of the pedestrians' vision (fields of view), and all the rules that applied, I have recorded the emergence of lane formations of different sizes in width, numbers, and distances between them. In general, the overall density needs to be above certain threshold values to observe lane formations. In this exercise, I used a 30 x 30 grid of hypothetically 65 to 80 cm, which is about the average width of a normal person's length of stride. In a population below 100 (overall density < 0.55 people-per-m²) I do not observe much lane formation since the pedestrians have almost no need to avoid others. In an overall density above 0.82 people-per-m², I started to recognize some clusters of pedestrians walking in the same directions. There is an issue of what precisely defines "lane." However, the main aim of this exercise is to understand the concept of this phenomenon, so whenever I observed more than two lanes of pedestrians flowing in the same directions, I decided to interpret them as lane formation. Next, I fixed the population at 450 (density close to 50%) and observed the difference with the various fields of vision (2.5m, 5m, 7.5m) and rules (2nd order rules On or Off) that applied (Figure 3.4). The rules that applied in the pedestrians' behaviors and algorithms have an order of applicability starting from their immediate neighboring

conditions (Moore neighborhood) to their larger surroundings. Some rules (rule order 2 and 3) are turned on and off based on each pedestrian's local density conditions (Figure 3.16). I found lane formation can be observed with or without additional rules, but they would affect the sizes and numbers of lanes. In this experiment, supplying more rules in addition to the basic rules, agents gain better capabilities to sense the surrounding conditions, and they form more efficient lane flows. Using more rules gives the agents behaviors more responsive to the different conditions. In conclusion, the more sensing capability pedestrians have, the wider lane widths become. Thus, the numbers of lanes also decrease, and eventually become two lanes – up and down directions.

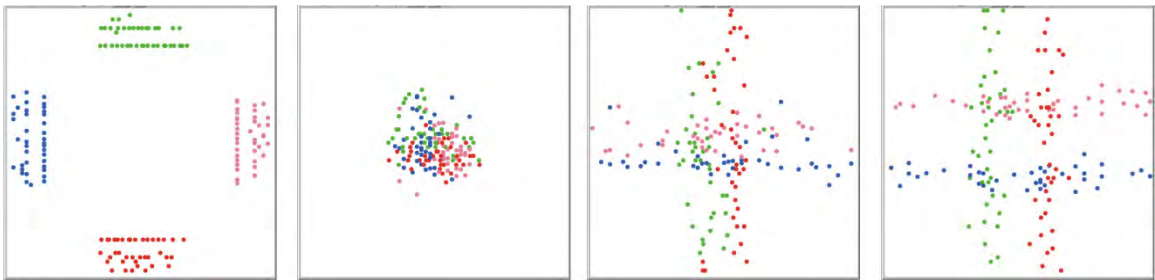


Figure 3.17 – Emergence of pin-wheel formation from simulation of pedestrian intersection.

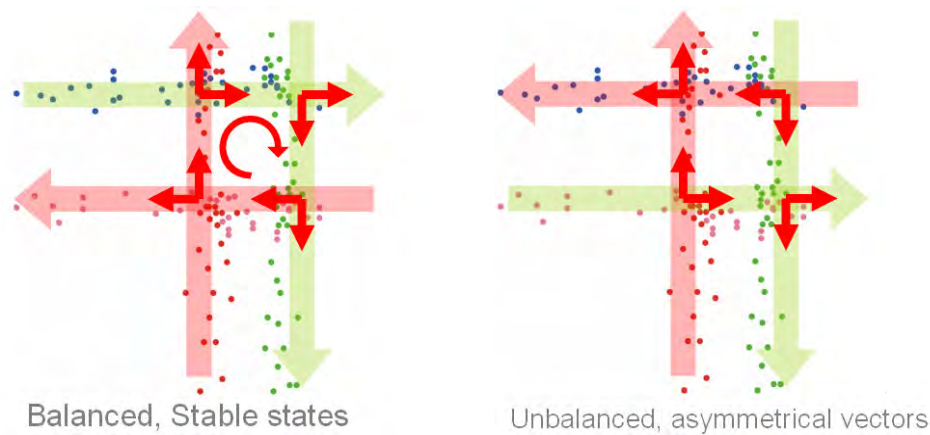


Figure 3.18 – Two possible directions for pin-wheel formations.

2) Pedestrian Intersection

I tested the same agent's behaviors on a scramble intersection (i.e., an intersection with diagonal crosswalks), often seen in urban settings in the Tokyo area. In this case, four groups of pedestrians are walking in different directions toward the intersection. The difference between the previous simple single-lane case and this intersecting condition is that there is no way to completely avoid all the collisions between the agents' groups moving in different directions. This time, pedestrians formed a pin-wheel-shaped formation instead of lanes. This is considered to be self-organizing collective behavior to minimize collision among them. Complete avoidance of collisions is not possible. Instead, the pin-wheel formation is their natural effort to minimize the points of collision down to four areas. Figure 3.18 shows two possible orientations for four lanes' flow directions. From the diagrams, it is comparatively clear that internally steadier formations evolve to balance the physics of flows. The above two exercises only implement locally defined individual behaviors, yet the resultant global behavior indicates self-organization among them.



Figure 3.19 – Simulation of Scramble intersection in Shibuya, Tokyo, Japan, using motion capture files for pedestrian movements inside urban modeling VR software.

3.2.6 Circle Packing

Another more intuitive example involves packing. When one is packing a large trunk or storage space with a number of boxes of various sizes, there is essentially no deterministic way to find the best packing solution inside the given space. However, by doing some trials, one will find it better to start with the boxes with larger sizes. Then, to fill the gaps with smaller boxes will yield the better solutions faster. This method does not give us a perfect packing solution, but guarantees decent optimal solutions faster.

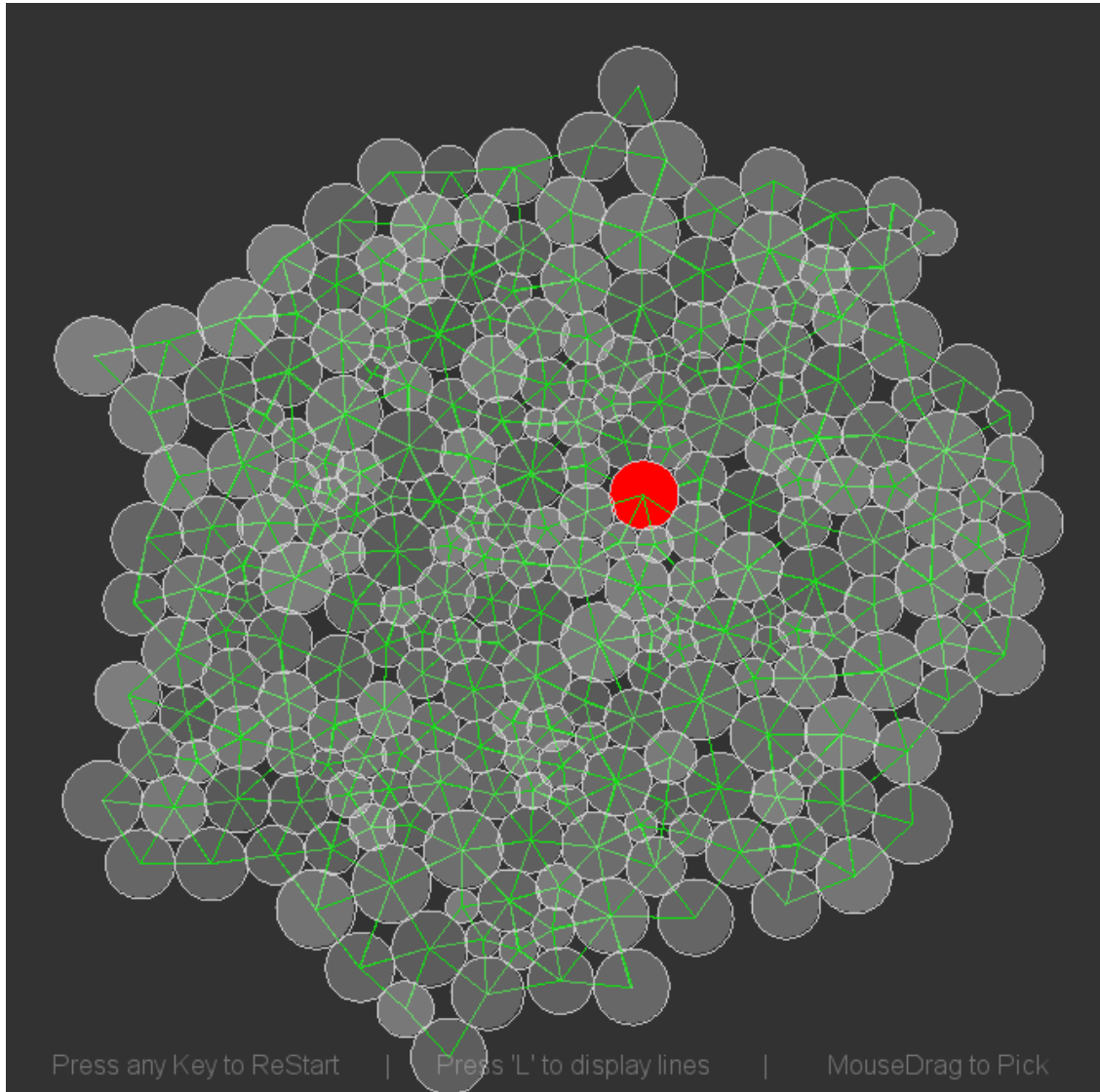


Figure 3.20 – Circle packing using a Bubble Mesh Method (Simulation by the author)

In mathematics, circle packing problems deal with arrangements of non-overlapping circles that fill a space. Close-packing of identical circles has been studied for many years, and some analytical solutions have been found for certain special conditions (Sugihara, 2006). However, even for numbers of less than a hundred identical circles, searching for the most compact packing of equal circles within a circle has not been resolved or is not known (Reis, 1975) – not to mention that the most efficient way to pack *different*-sized circles within a circle is not known at all. So, usually, we seek for optimized solutions, instead of the best solution, using heuristic techniques. This problem's primary condition, non-overlapping of circles, is a non-linear problem, so it is not easy to solve. Even using the heuristic optimization techniques, it requires extended time for calculations.

Simulation is another completely different approach to solving this problem. We can consider physical disks inside a larger circular container that can push and pull each other until they find their stable configurations. Dynamic behaviors of disks can be computationally simulated using a bubble meshing method. Bubble meshing is a technique used for Finite Element Method (FEM) to obtain uniform subdivision of finite elements from a given entire domain. The method uses close packing of bubbles for the optimization of mesh node locations. Two adjacent bubbles literally push and pull each other using a repulsive or attractive force similar to an intermolecular van der Waals force. A globally stable configuration of tightly packed bubbles is determined by solving the equation of motion.

Each circle has simple behaviors – push and pull – and the interactions of circles with locally implemented behaviors eventually produce globally functional solutions. This

gradual, but emergent formation process is not externally imposed by any information. Aggregation of simple entities with active behaviors can produce globally meaningful results through local interactions.

“Build a system composed with active primary entities, and let them find design solutions that satisfy their current environmental conditions” is not a common attitude toward design. However, we are starting to see these examples from many natural systems’ behaviors, and some human-implemented systems models. These are new opportunities for us to consider new types of design decision-making systems that can be more adaptive to changes. In other words, instead of providing a concrete picture of design as a final product, we might be able to provide simply a process or logic of formation that can lead to a design that conforms to the better solution in a spatiotemporal manner. In this thesis, I would like to investigate the potentials of emergent methods in our design activities.

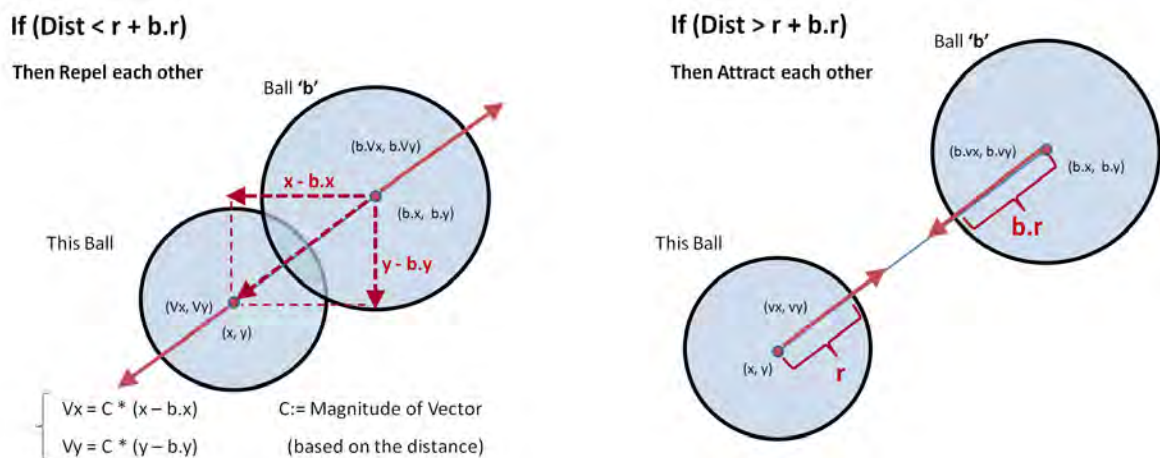


Figure 3.21 – Bubble mesh method: “push” and “pull” behaviors for circles.

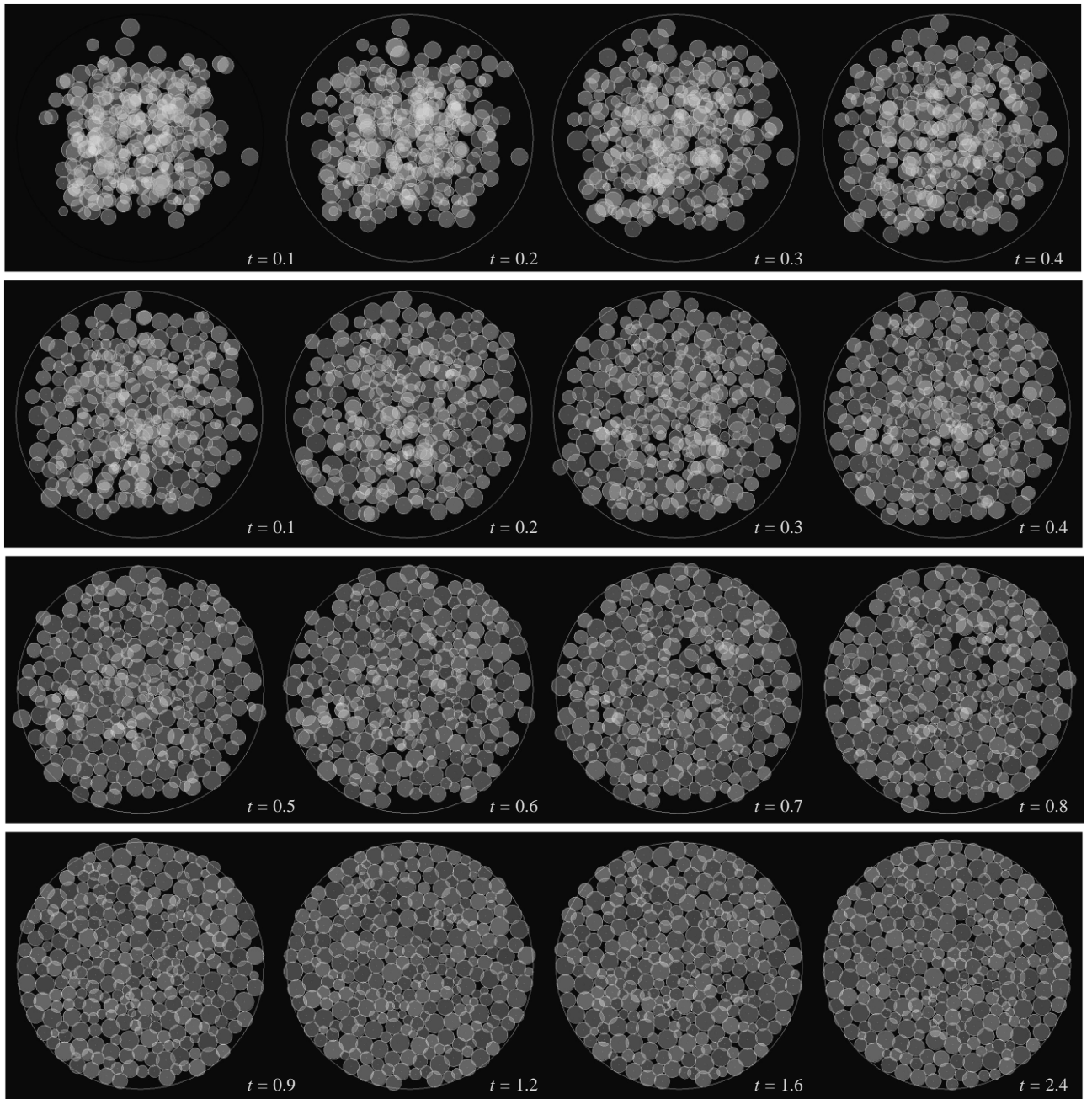


Figure 3.22 – Process of Bubble Mesh Method: Close packing of 400 different-sized circles.
(Simulation by the author)

3.3 Concluding Note: The Trans-dimensional Topology Concept

I have explained the evolution of three stages of computational methods for design, starting from *evaluation* methods, which can only evaluate given solutions, and moving to *design search or generation* methods that can search and select solutions from given conditions using the evaluation methods. In case of some algorithmic methods such as GA's, well-defined evaluation methods such as fitness functions are necessary for their mechanisms to perform as design search methods, and evaluation and design search methods are in a dependency relationship.

The second and the third categories of computational methods – *design search* and *growth + adaptation* – are rather independent of each other. Self-organizing computation has been introduced as one approach to implement the *growth + adaptation* method. Self-organizing computation normally presents one result of spatiotemporal structure at a time, and there are some possibilities that every result may differ due to its characteristics – *randomness and amplitude of fluctuations*. In order to search out optimal results from self-organizing computation, these different results may need to be evaluated by another selection sequence. Populations of different growth patterns gained from self-organizing computation can be used for initial inputs for design search methods to find optimal growth patterns. Thus, the second and the third methods can be used in tandem.

The third category – *growth + adaptation* methods – are equipped with a feedback system between the design context and objects being designed. This co-evolutionary process can only be represented through space + time four-dimensional topology space;

however, in my opinion, four dimensions are not enough for successful implementations of this model. *Trans-dimensional topology* is an alternative conceptual viewpoint that I propose as a concluding note for this chapter. In order for us to think of design in a spatiotemporal manner, design thinking inside the trans-dimensional topology world is suggested. We tend to think only in three-dimensional spaces as our design problem, and currently available technologies expect us to frame a solution within a static time frame.

Trans-dimensional topology space is a five-dimensional hyperspace that contains three dimensions for a Cartesian coordinate system, one dimension for a time axis, and one additional dimensional axis for all possible alternative solutions. This final axis contains all possible sequences of designs in time series. Here, we are dealing with a type or category of designs that can recurrently alter their morphologies in response to external environmental changes. I propose systems thinking that resides inside the parallel worlds of alternative schemes. Some schemes can grow into different schemes and branch out to form topologically continuous yet spatially separated schemes. Some branches in trans-dimensional topology space may merge back again during the course of evolutionary growth in the four-dimensional world of space + time.

Searching for solutions within this space is overwhelmingly exhausting work not only for humans but also for computers, as it requires extensive parallel thinking. Under this condition, conventional analytical means are likely to be in vain, and instead, bottom-up design approaches become potentially preferable. As I have reviewed in this chapter, heuristic approaches seen in close packing of circles and lane formations by pedestrian agents are good examples that represent shifts in methodology from analytical to heuristic.

In the next few chapters, I would like to explain several architectural applications of this logic.

Chapter 4

Experiments in Evaluation and Selection (Design Search)

Introduction

In next two chapters, implementations of the aforementioned three stages of computational methods into various design problem frameworks are introduced. The first example uses a technique similar to that we reviewed from a bubble meshing method using dynamics of physics. This example falls into the first category of computational method – evaluation; however, later, this framework will allow us to actively find optimized solutions. Globally stable structures, such as catenary shapes, can be progressively gained through locally defined behaviors of particle masses based on explicit time stepping.

The second example falls into the second category of computational method – search. This example uses the principle of evolutionary algorithms to search the new geometrical structures using techniques from turtle geometry and L-system. An architectural implementation using various different fitness measures is introduced.

4.1 Implementing Physical Reactions to CAD System

In this section, I would like to introduce a new kind of CAD system that allows users to intuitively see the physical forces and their reactions acting on the structures. Use of dynamically animated structural deformation informs balances of structures on the schematic level while users are modeling them in a real-time manner. This system resolves all forces acting on a structure into axial-forces in order to approximate and visualize structural behaviors using Finite Difference Method (Euler Method). In addition to this analytical capability for static structures, the software allows users to design kinetic components. Users can assign actuators that can vary their lengths in a certain frequency. Kinetic architecture can be generated from this platform. Explanation of this implementation in detail follows in the next section.

This system has a dynamical mechanism that can deliver globally stable structures such as catenary shapes through locally defined interactions of physical forces at particle masses. Although the system is primarily designed for evaluating structural performance, the system's ability to develop macroscopic spatiotemporal structures out of processes and interactions defined at the microscopic level exhibits the typical feature of self-organizing systems. Due to this characteristic of the form-finding process, this method was included in this chapter. In this case, the system's behaviors are rather deterministic,

compared to several other systems that follow in this thesis, since the method does not rely on any type of randomness or amplification of fluctuations.

4.1.1 Elastic Spring Mass Object:

In order to simulate dynamic reactions and movements of two-dimensional surfaces or three-dimensional solid objects with reasonable calculation speed of the computational environment, surfaces or solids are subdivided into discrete cells or voxels. The resolutions of this subdivision can be controlled by users. The finer the subdivision, the more accurate and precise the details and behaviors – that is, closer to the original solid objects – that can be obtained.

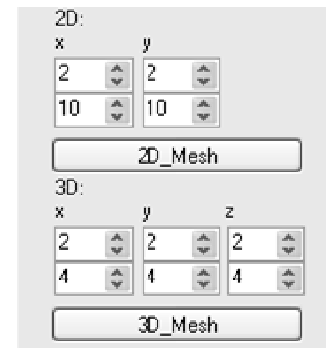
User Interface controls the following parameters.

2D Elastic Spring Mass (Surface Membrane, Cloth)

- Set x-y dimensions.
- Set numbers of divisions for x & y directions
- Generate 2D Spring Mass object

3D Elastic Spring Mass (Solid Object)

- Set x-y-z dimensions.
- Set numbers of divisions for x, y, z directions
- Generate 3D Spring Mass object



The user interface is divided into two sections: 2D and 3D. The 2D section has input fields for 'x' (value 2) and 'y' (value 10), with up/down arrows, and a '2D_Mesh' button. The 3D section has input fields for 'x' (value 2), 'y' (value 4), and 'z' (value 2), each with up/down arrows, and a '3D_Mesh' button.

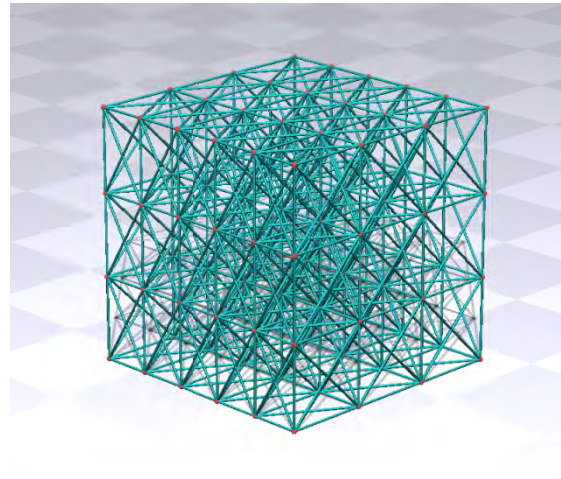
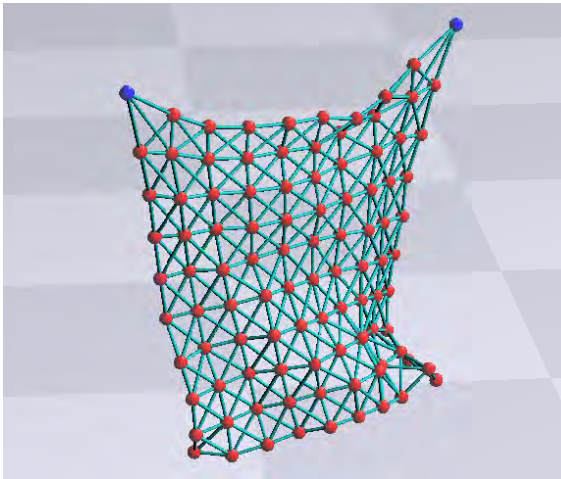


Figure 4.1 – 2D SpringMass w/ 10x10 divisions (left). 3D SpringMass w/ divisions x=5, y=3, z=3 (right). User Interface of the program (above).

This is a well-accepted approach in continuum mechanics which assumes that the matter in the body is continuously distributed and fills the entire region of space it occupies. In continuum mechanics, certain phenomena can be modeled by continually subdividing a body into infinitesimal elements with properties that are those of the bulk (original) material. In the case of this experiment, surfaces or solids are replaced by interconnected particle masses using the vertices of subdivision, and elastic properties of materials under consideration for the body are approximated by the interconnected network of virtual springs with appropriate values for spring constants. The original material's Young's modulus of elasticity is represented by these interconnected springs with certain force constants behaving according to Hooke's Law. This process turns the complex forces acting inside the continuously dense original objects into axial forces and reduces computational time. In the case of a two-dimensional surface, a result of this process becomes the simulation of cloth or surface membranes. Interconnected springs at each subdivision require diagonal members as a shear spring to properly simulate the physical characteristics of cloth. In the case of a solid object, this process will produce a three-dimensional truss structure using the original object's shape outline.

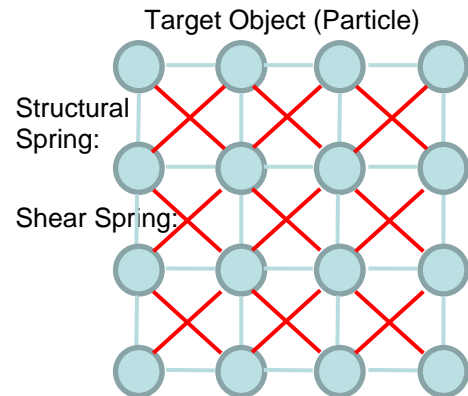
Concept of Elastic Spring Mass System

- Elastic Object Class Structure

```

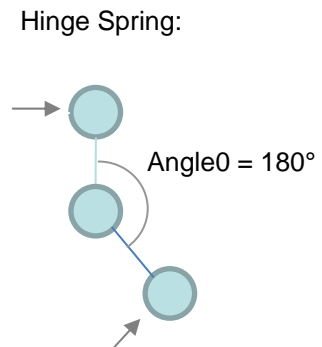
Elastic E = new Elastic( );
{
    Target Particle[16]
    Spring Struct[32]
    Spring Shear[18]
    Spring Hinge[12] (only for 2D)
    (Invisible hinges that try to maintain initial angles)
}

```



- Acting Forces

K-value: Spring constants for Structure, Shear, and Hinge springs (3000kg/m, etc.)
Damping: damping coefficient for springs
Mass: m = Mass for Particles
Gravity: $g = 9.8\text{kg/m}^2$
Drag force: (relative to particles' velocities)
Repel factor: Coefficient among particles
Elastic collision const: (between floor & particles)
Friction: (between floor & particles)



Forces:

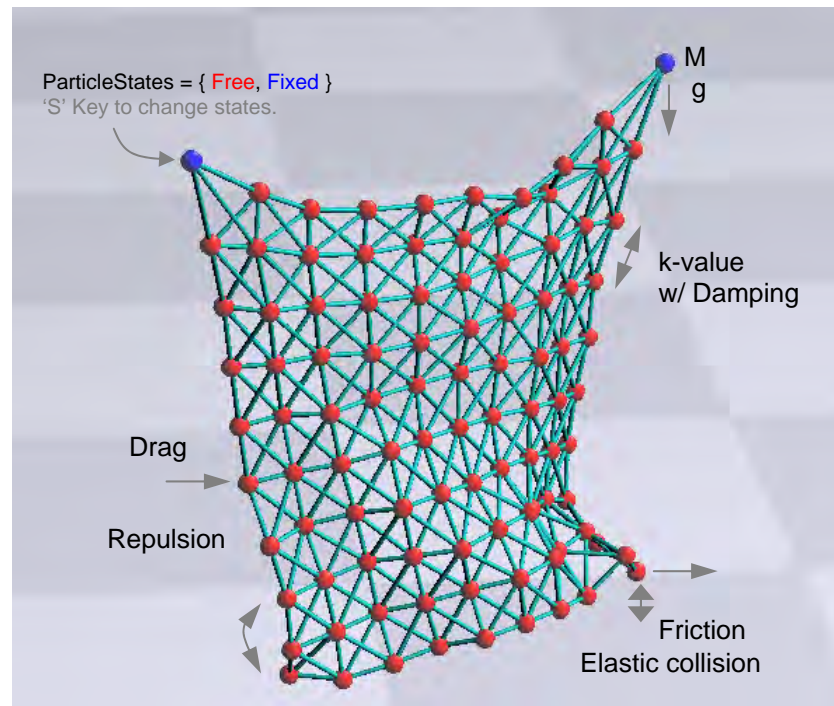


Figure 4.2 – Various forces acting on the 2D structure.

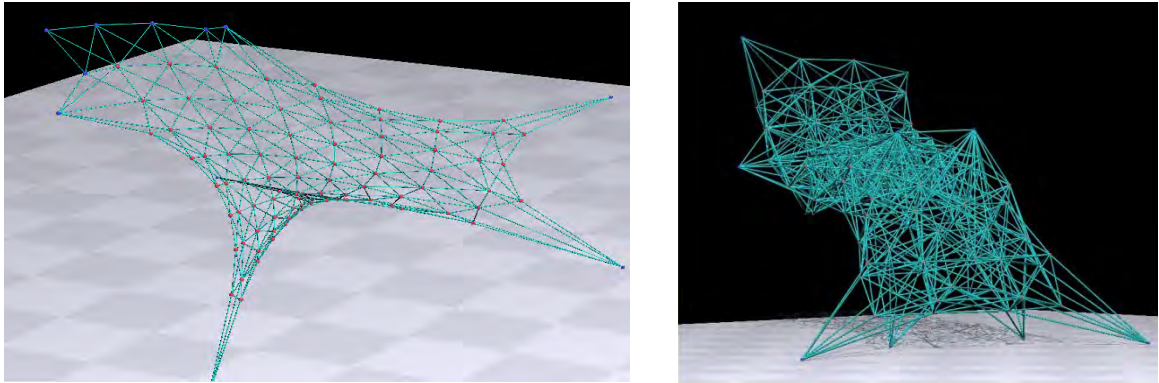


Figure 4.3 – Model Examples using Elastic Spring Mass System

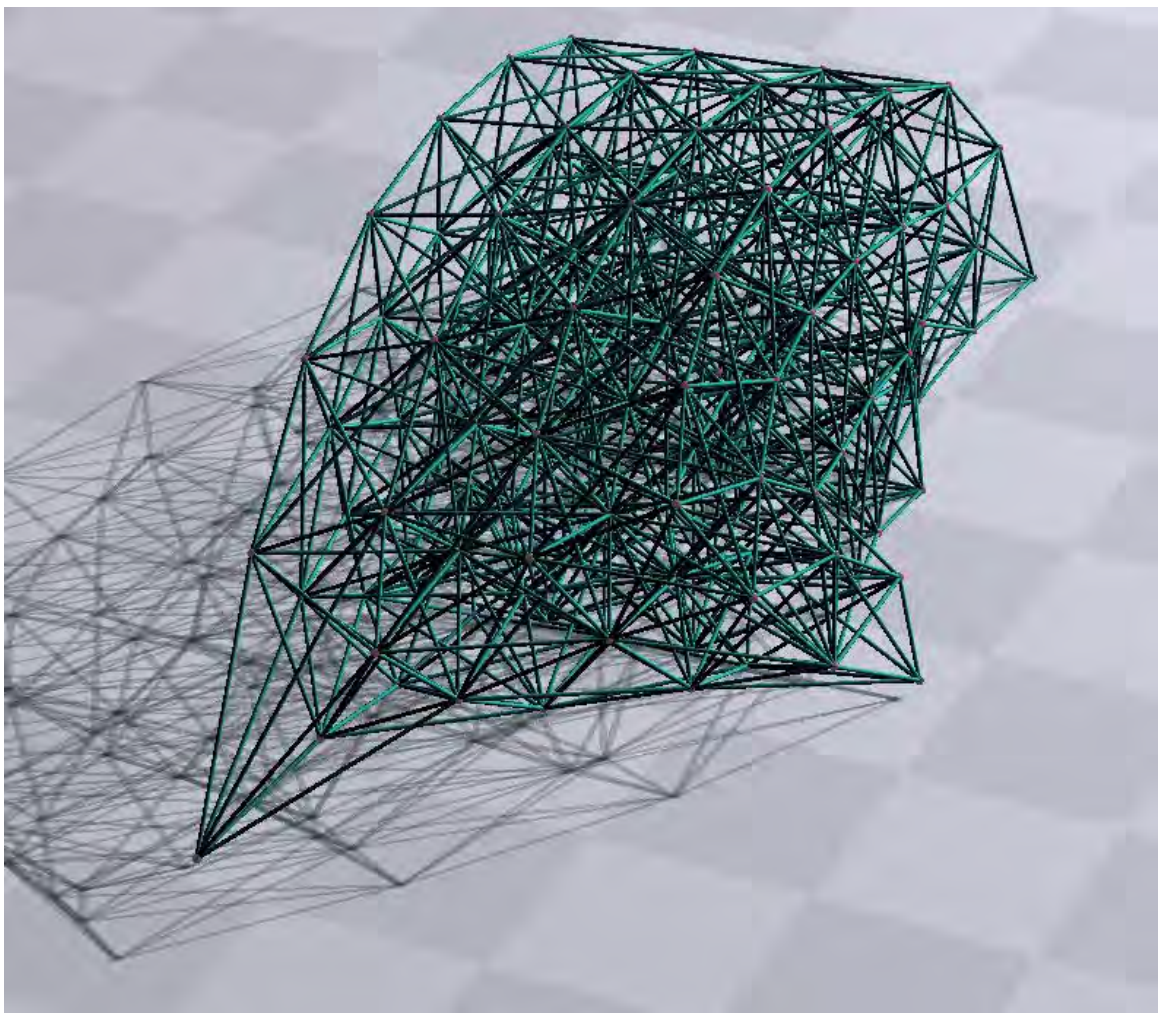


Figure 4.4 – Catenary Shapes can be gained from the use of the software. Geometries of Stressed-skin Membrane structure and thin shell structures can be derived intuitively at the schematic level.

4.1.2 Finite Difference Method: Explicit Time Stepping

This technique allows representation of animated structural deformations by stepping through the displacements of particles in *delta-t*, finite time segments.

Forces on springs are derived from Hooke's Law.

$$F(t) = k * x(t) \quad (\text{Hooke's Law})$$

From Newton's Law of Inertia, the time derivative of the momentum is gained.

$$F(t) = M * A(t)$$

$$A(t) = \frac{V(t + dt) - V(t)}{dt} \quad (dt \rightarrow 0)$$

$$V(t) = \frac{x(t + dt) - x(t)}{dt} \quad (dt \rightarrow 0)$$

Using 1st and 2nd Taylor polynomials, calculate and update current positions of particles.

$$A(t) = \frac{F(t)}{M}$$

$$\begin{aligned} V(t + dt) &= V(t) + A(t) * dt \\ x(t + dt) &= x(t) + V(t) * dt \end{aligned}$$

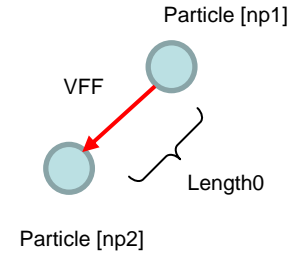
This method approximates the trajectories of movements of particle masses. The algorithm uses explicit finite time steps, *delta-t*, to move each mass forward by calculating a displacement from the above equations. (See code sequence below.) The algorithm iterates through this calculation for displacement for each mass through all springs so that the displacement at one location is translated through all other locations of an object, and eventually, it describes motion of the entire structure.

Other primary forces under consideration are friction between a floor and particle masses, damping coefficient for springs, drag force relative to particles' velocities, repel factor (Coefficient among particles), and gravity (9.8kg/m²). Damping forces are considered, because otherwise springs never slow down their oscillations. The drag forces between

particles are added in order to avoid overlapping of surfaces or solids. These forces are applied when calculating forces on each particle. All forces from adjacent particle masses are translated and added to the forces affecting the particle mass under the calculation.

- Calculate Forces on each Particle (Iterate through all Springs)

```
for(i = 0; i < numSPRING; i++)
{
    vDir1 = Vector3.dirc(particle[np1].vPos,
    particle[np2].vPos);
    len    = Vector3.dist(particle[np1].vPos,
    particle[np2].vPos);
    vFF    = (structK*(len-
    structS[i].length0))*vDir1;
    particle[np1].vForce += vFF;
    particle[np2].vForce -= vFF;
}
```



- Finite Difference Method (Explicit Time Stepping)

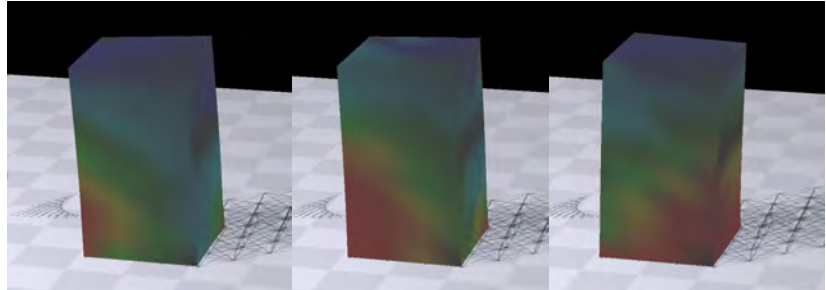
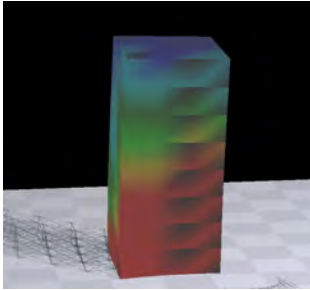
```
for(i = 0; i < numParticle; i++)
{
    if(particle[i].flagFixed == true) continue;

    // Acceleration
    particle[i].vAccel = particle[i].vForce / massParticle;
    // Velocity
    particle[i].vVelocity += particle[i].vAccel * dt;
    // Displacement
    particle[i].vPos += particle[i].vVelocity * dt;
}
```

4.1.3 Stress Display: Animated Dynamic Structural Deformation

The software can dynamically display real-time stresses acting on each particle node. Axial stresses at each node are rendered with graduation of colors: Red being high stress to blue being low stress. The color range of stress level can be adjusted by a stress gauge. This feature can intuitively inform designers about areas of stress concentrations in a real-time manner. Later in the second and third sections of this chapter, this same functionality is used for visually evaluating structural performances of design schemes. In the third section, housing layouts created by DLA (Diffusion-Limited Aggregation)

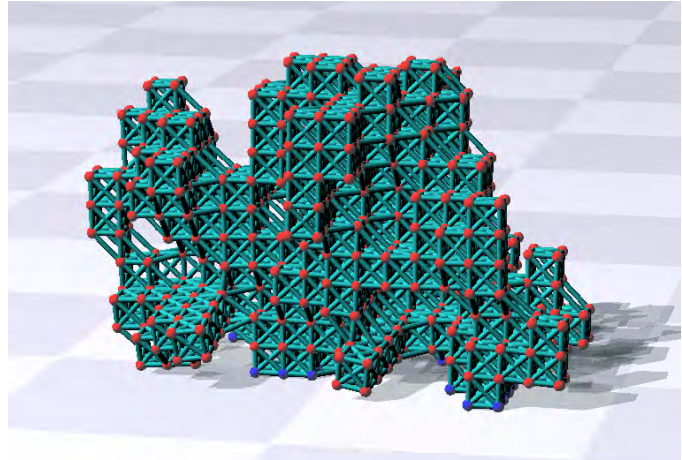
algorithm are tested with this method, and forces are calculated based on axial-forces within the three-dimensional truss structure. (See later sections.) In the second section, this feature is used as one of the fitness functions for the evolutionary algorithm.



Press 'space'-key for Stress Display for Solid objects. Range of stress level can be adjusted by a Stress Gauge.



Housing layout created by DLA (Diffusion-Limited Aggregation) algorithm. Forces are calculated based on Axial-forces on 3D-truss structure. (top-right)



Inform designers intuitively of areas of stress concentrations in real time. (bottom-right).

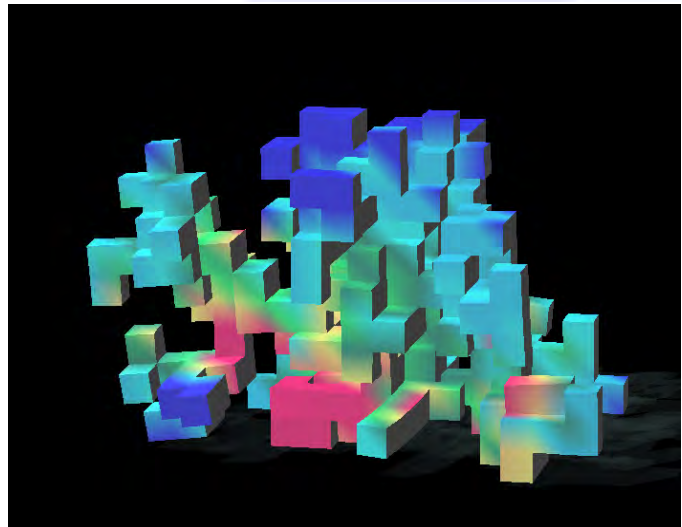


Figure 4.5 – Software can dynamically display real-time stress level based on the stress at each particle divided by numbers of particles. Red means high stresses to blue means low.

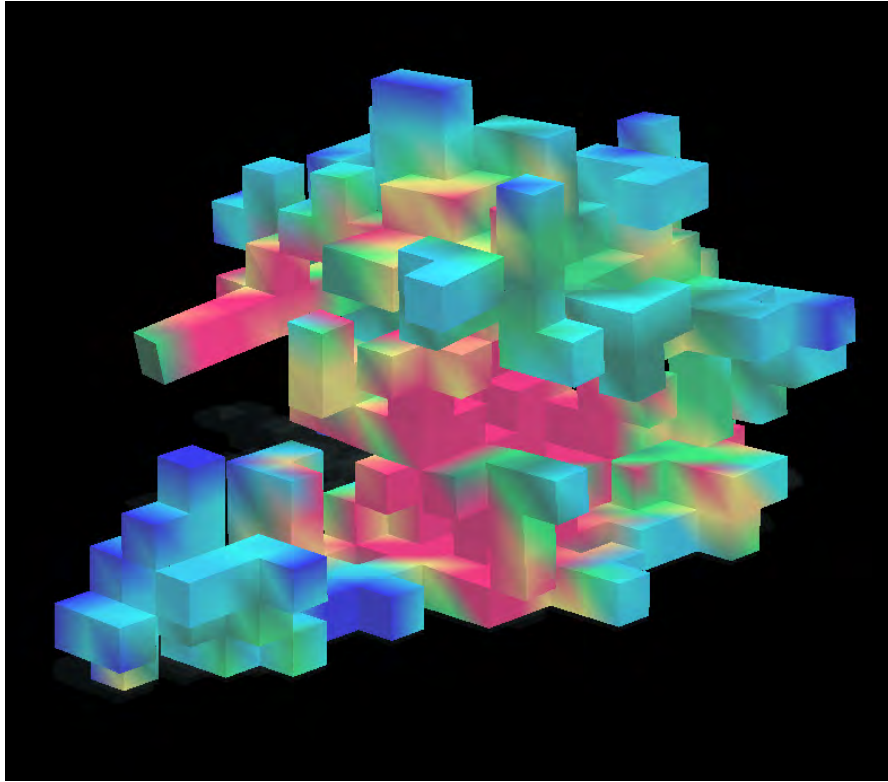


Figure 4.6a – Animated Deformation Sequence_1

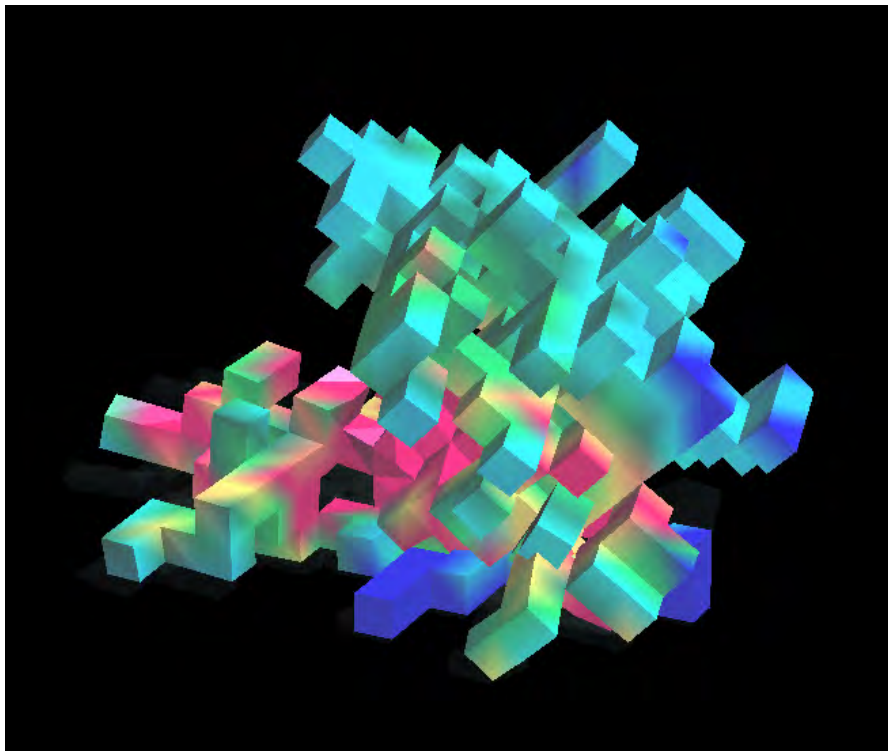


Figure 4.6b -- Animated Deformation Sequence_2

4.1.4 Kinetic Objects

Kinetic characteristics of various objects can be implemented in this program. This program allows users to replace conventional spring members with oscillating pistons. Users can also add new piston members to existing structures, and set a length of member, amplitude of actuation, a period of gait cycle, and a phase of oscillation. This feature allows users to actively construct members for kinetic architecture. For example, a variable-length truss system or tensegrity system can be designed in this environment. Orchestrating kinetic members' gait cycles and amplitudes has become a challenge as synchronization of their movements as a whole becomes a complex issue. New kinds of design methods are expected for the design of active objects.

- Code Implementation of Piston:

```
//PISTON
Form1.time=(Form1.time + dt/10)*(2*Math.PI);
if(!pist.Contains(i))
    vFF=(structK*(len-structS[i].length0))*vDir;
else
    vFF=(pistonK*(len -structS[i].length0/3*Math.Cos(Form1.time
    + i* Math.PI/(numParticle+1))-structS[i].length0/3*2))*vDir;
```

- Mathematical Implementation of Piston:

$$L := L_o + A * \cos\left(\frac{t}{\lambda} + \varphi\right)$$

L := current length

L_o := original length of segment

A := amplitude of actuation

λ := period of gait cycle

φ := phase

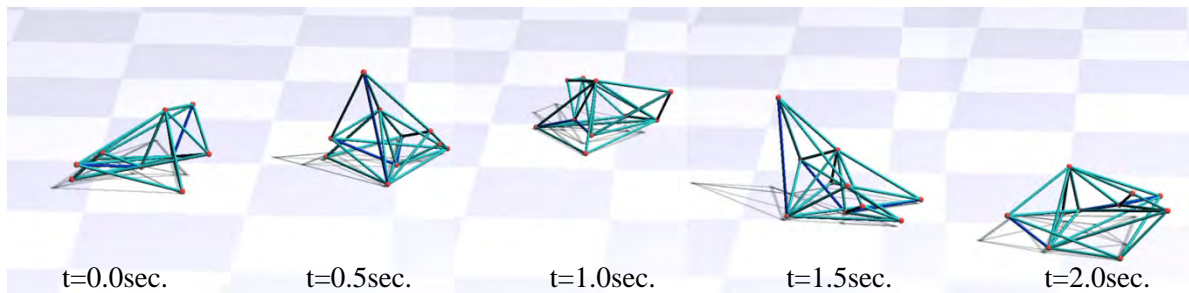


Figure 4.7 – Piston Segment (blue) can be added to structures. The system can create randomly generated segments with random numbers of Actuators.

- Design Platform for Kinetic Architecture (Draw Mode)

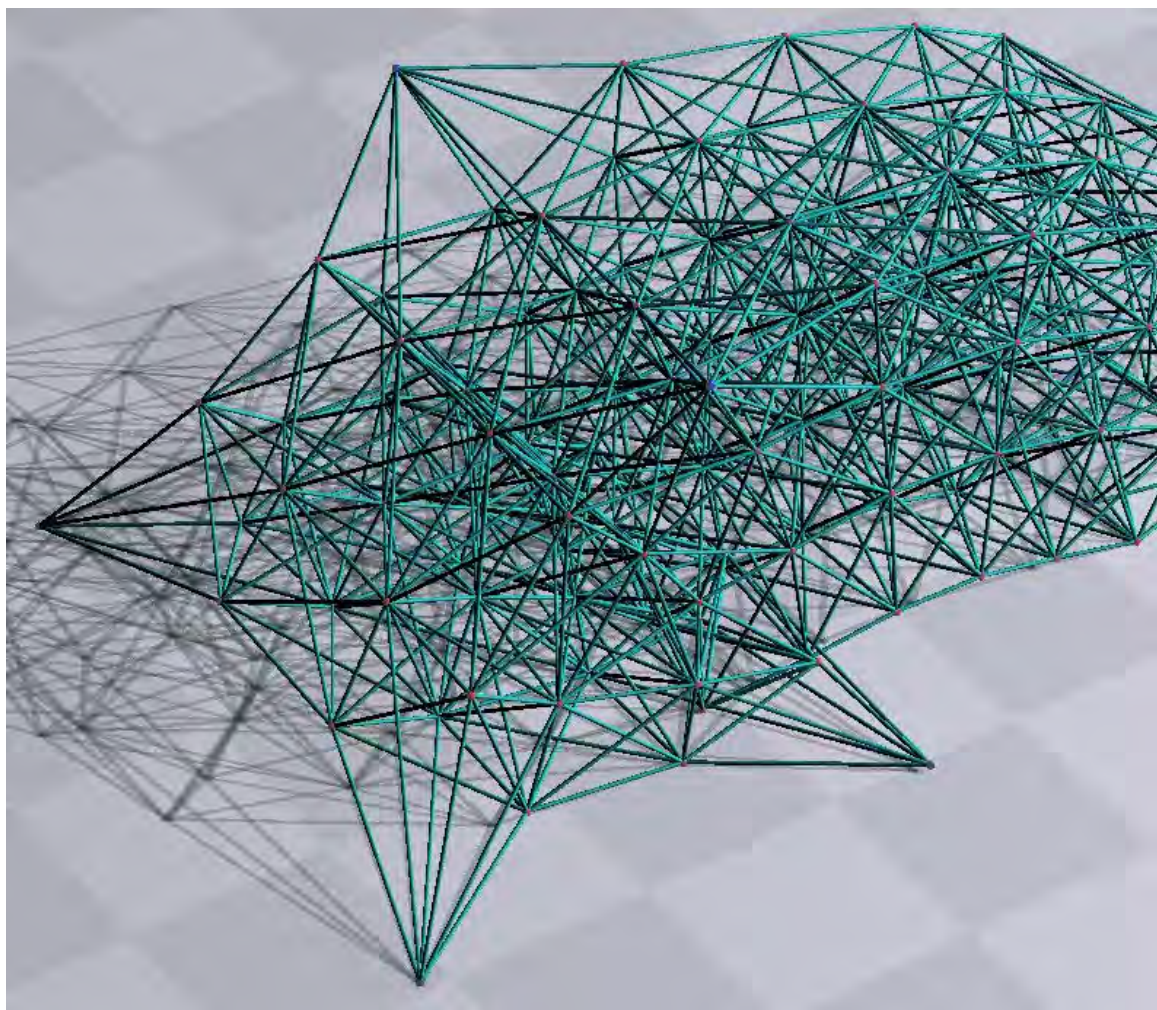
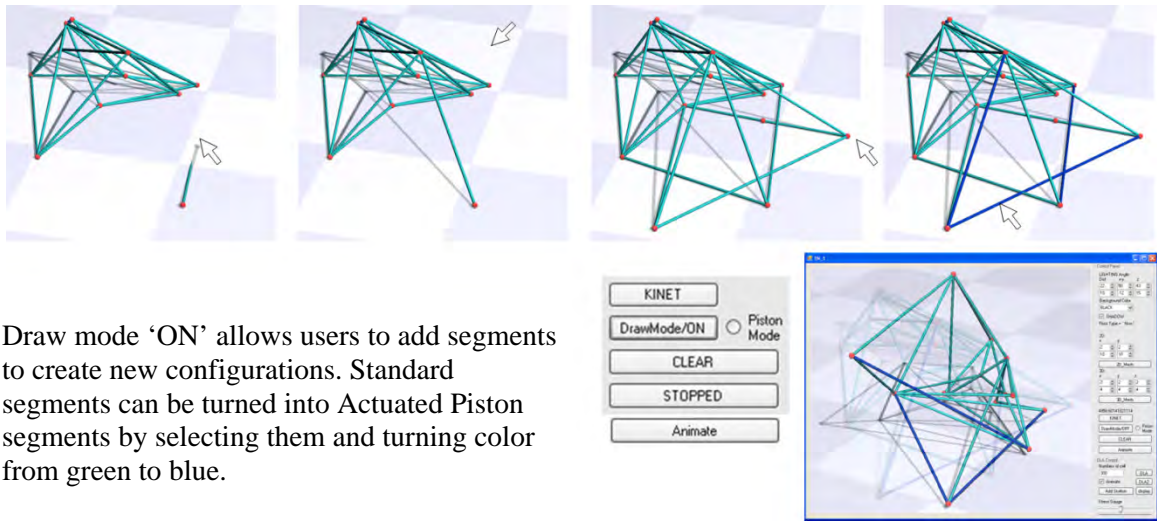


Figure 4. 8 – Example of Kinetic Structure (w/ Animated Dynamic Structural Deformation)

4.2 Design Search: Turtle Implementation of L-system

4.2.1 Introduction

In this section, I would like to introduce a system to create generative designs and represent implemented functionalities of the software. The concepts discussed in the following section are inspired by Lindenmayer systems, introduced by Aristid Lindenmayer and Przemyslaw Prusinkiewicz (1990), and ideas of turtle geometry introduced by H. Abelson and A. diSessa (1982). Integration of turtle geometry and L-system was also introduced in (Lindenmayer, 1990) and in more recent work by H. Gregory (2003). H. Gregory displayed the use of these methods for various design purposes, including a design of tables from a few utility functions. A number of papers on evolutionary computation from the area of computer science, including (Gregory, 2003), have become an inspiration for this project. However, their papers do not address the architectural applications of generative design systems. One of the main goals of this thesis is to study the potential use of generative design algorithms for architectural design at the schematic level. This study will seek a possibility of using pure mathematical and computational strategies in design creation processes.

4.2.2 Method

The generative design system is composed of a part that generates various designs, the building engine, and a part that evaluates various design instances, the selection sequence. This framework follows the second method introduced in the last chapter – *search*. The former part of the system, *generation*, uses a turtle interpretation of L-system, and the latter, *selection*, uses evolutionary algorithms based on genetic algorithm/programming.

Design instances created by the former system are evaluated by the latter system to form better instances for the next generations, and this iterative loop produces a computational selection process analogous to natural selection.

4.2.3 Design Generator (Building Engine)

The advantage of using L-system with turtle geometry is that the design can be described by procedural (functional) representation composed of series of strings instead of using hard-coded geometrical information. This will allow for the representation of geometric shapes to be more concise. It is also easier to extract their geometrical characteristics in abstract form and to edit them more efficiently.

The turtle works as a builder for designs, and ‘*string*’ of command sequences gives instructions to a turtle to build structures out of unit voxels. The table below shows different types of instructions that form the design language for the turtle. The turtle places voxel building blocks as it moves based on the instructions composed of strings.

The turtle works as a builder agent in this application.

Code	Description
F(n):	move in the positive x direction n units
B(n):	move in the positive x direction n units
R(n):	rotate heading about z-axis $n \times -90^\circ$ (right)
L(n):	rotate heading about z-axis $n \times 90^\circ$ (left)
U(n):	rotate heading about y-axis $n \times 90^\circ$ (up)
D(n):	rotate heading about y-axis $n \times -90^\circ$ (down)
C(n):	rotate heading about x-axis $n \times 90^\circ$ (clockwise)
O(n):	rotate heading about x-axis $n \times -90^\circ$ (counter clockwise)
[](n):	repeat enclosed operations n times (replicator block)
{ }(n):	restore orientation & location (push & pop operator)

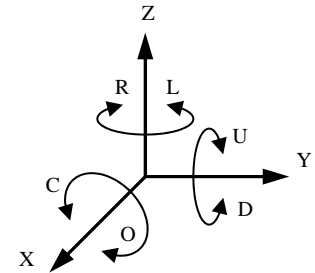


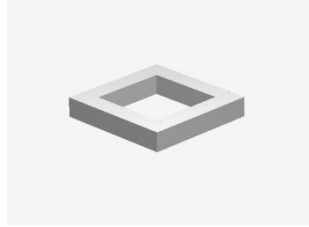
Table 1: Design Building Language

Builder's Direction in 3D

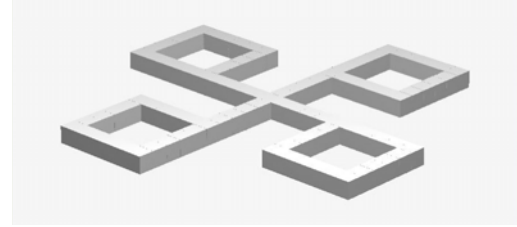
- Generative Grammar Instruction



F5R1F5



[F5R1]4
= F5R1F5R1F5R1F5R1
Using Replicator bracket


$$\begin{aligned} & [\{ F_5(F_5R_1)4 \} R_1] 4 \\ & = \{ F_5F_5R_1F_5R_1F_5R_1F_5R_1 \} R_1 \{ F_5F_5R_1F_5R_1F_5R_1F_5R_1 \} R_1 \{ F_5F_5R_1F_5R_1F_5R_1F_5R_1F_5R_1 \} R_1 \{ F_5F_5R_1F_5R_1F_5R_1F_5R_1F_5R_1 \} R_1 \end{aligned}$$

Using Push/Pop Matrix bracket to restore Turtle Position

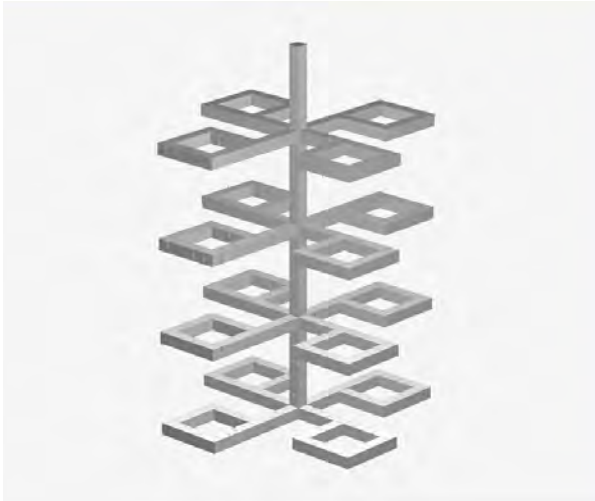
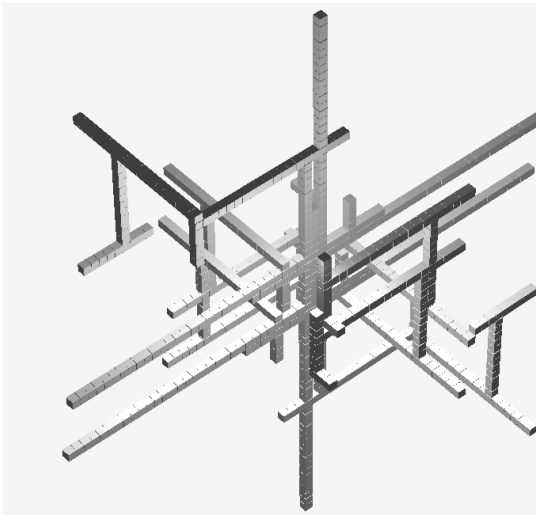
[[[F5[F5R1]4]R1]4U1F8D1]4
=.(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1U1F8D1(F
5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1U1F8D1(F5F5
R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1U1F8D1(F5F5R1F
5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1(F5F5R1F5R1F5R1F5R1)R1U1F8D1[illegible]

Figure 4.2.1 – Examples of design pattern using Generative Grammar.

4.2.4 Generative Grammar Instructions

L-system is a grammatical rewriting system that can encode various geometrical schemes. A hypothetical virtual movable node is commonly called a “turtle” in the field of computer graphics, and a turtle’s trajectories are represented by a few alphabetical characters which represent the turtle’s heading directions and movements. For example, ‘F5’ means that a turtle moves forward for 5 units’ distance, and ‘R2’ means that a turtle turns 2×90 degrees right from a current heading direction. These alphabets and numbers can form sentences that instruct a turtle to move to a specific location. In this program, I use turtles’ trajectories to produce three-dimensional structures. In principle, combinations of these operators and numbers can form any contiguous geometry in three-dimensional lattice space. In addition to these basic operators listed in the above table, there are two special operators: [] – the replicator operator, and { } – the push-pop operator.

The replicator operator repeats a block of operations enclosed by square brackets for a given number of times that is indicated right next to the brackets. (See the examples below.) The push-pop operator restores a current location of a turtle while it executes operations within curly brackets. After the execution, the turtle goes back to the original location and starts executing the operators preceded by the curly brackets. This operator is useful when one is designing branches.

- Generates String in Random Sequence

```
string[] commands = new string[] { "F", "B", "F", "B", "F", "B", "L", "R", "U", "D", "C", "O" };
int leng = rand.Next(randomLength)+codeLength;
for (int i = 0; i < leng; i++)
{
    code += commands[rand.Next(12)];
    code += rand.Next(9).ToString();
}

//Generated Code sequence
code="F2R5L4F8B3O3F2C1U3F8L2B3D4B0O0B5O8F4D0D1F0F5O0U1F1F6U7O8D1D8F3F3F3B4B3
F5R0L3F7B6C0R6U5C0O1U5F7O0F5R5R1B0D6B6B4D8D1B1D2U4L3L8R6B7F4F5F7F5O4O0B1U3F7"
```

- Inserts Replicator Brackets and Push/Pop Operators ([] & { })

```
code="F2R5[L4F8{B3O3F2[C1U3F8L2B3D4{B0O0{B5O8F4[ [D0D1F0F5]3O0U1F1F6U7O8D1D8F
3]4}F3F3{B4B3F5}}R0L3F7B6C0R6U5C0]4O1U5{F7O0F5R5R1}B0D6B6B4D8}D1B1D2U4L3]4{L
8R6B7F4{F5}F7F5O4O0}B1U3F7"
```

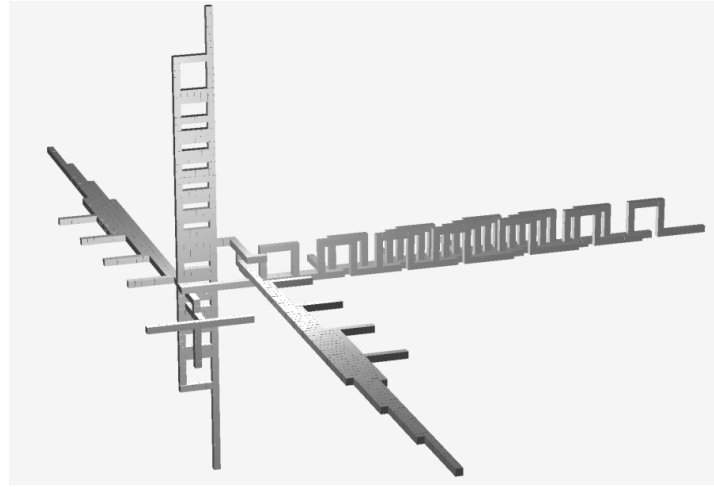


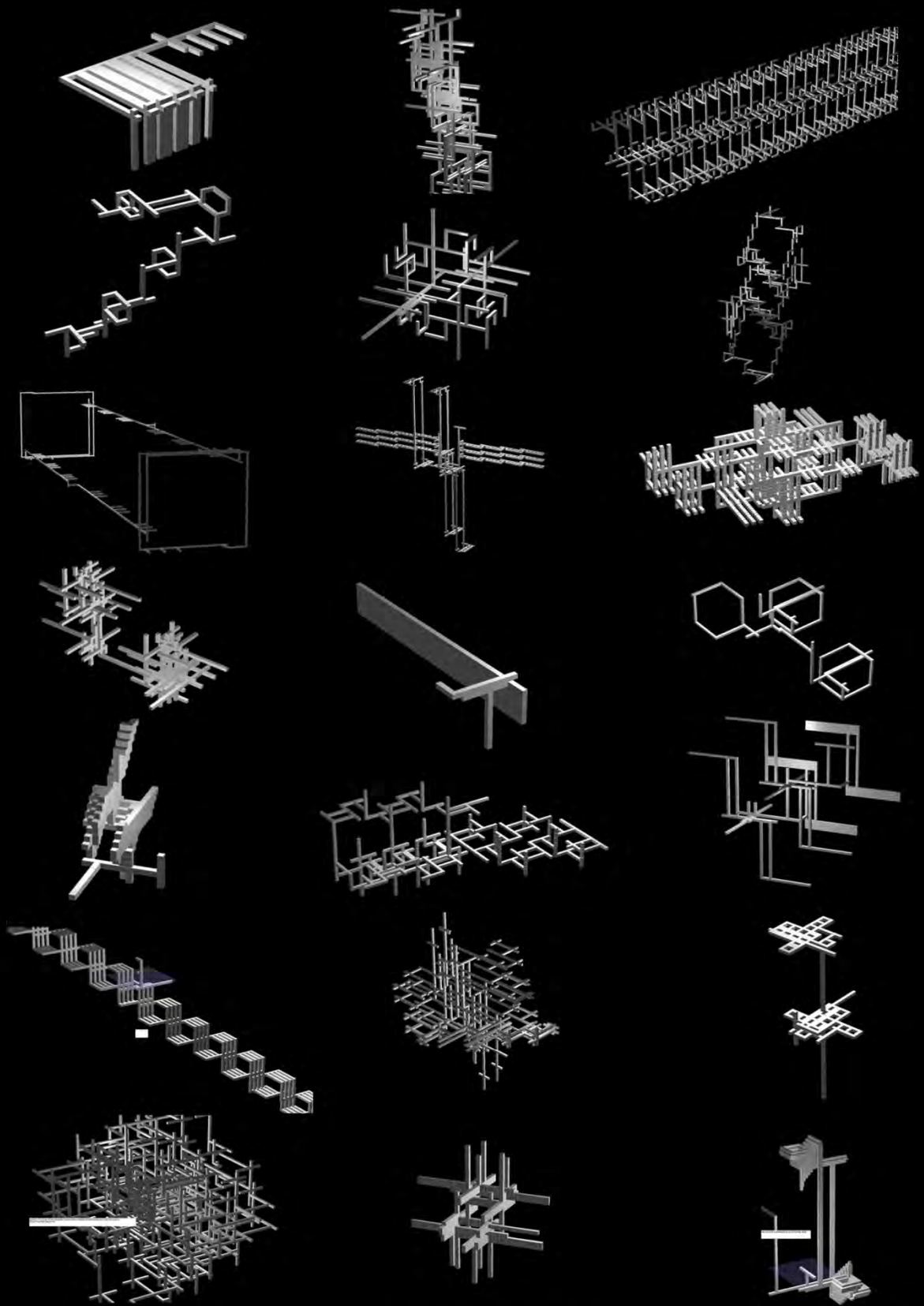
Figure 4.2.2 – Example of Auto-generated design pattern from the code above.

4.2.5 Evaluation Sequence Using Evolutionary Computation

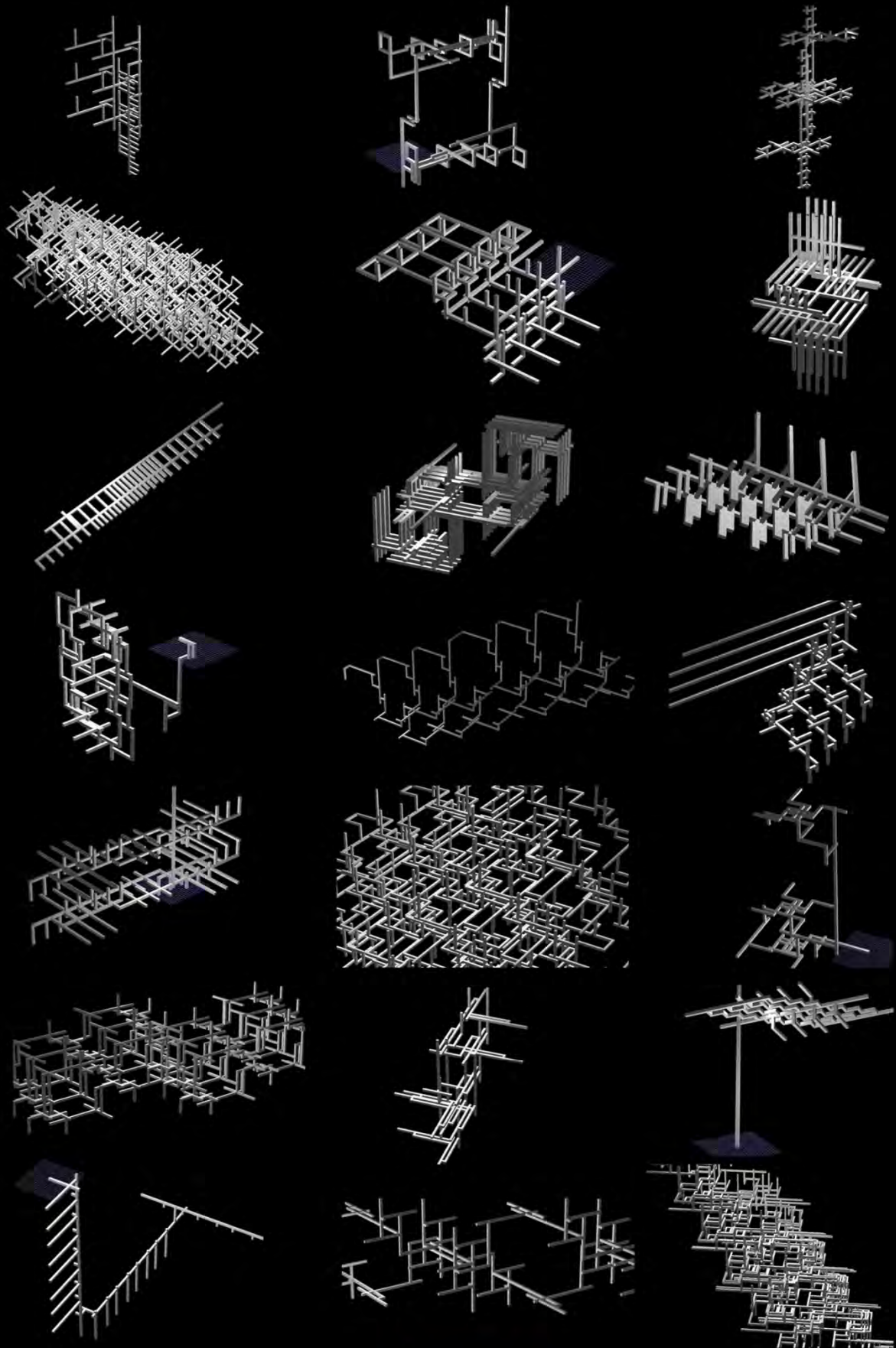
This program's design search process follows an algorithmic flow of a genetic algorithm. After making an initial population randomly, individuals (schemes) in the initial population are evaluated based on various architectural fitness functions. Elite schemes remain as parent schemes to generate new generations of population by using operations such as crossover and mutation. The algorithm repeats the same sequence until it satisfies required fitness threshold values. Encoding of geometrical information using turtle geometry and L-system is useful because it permits string-based operations such as crossover and mutation.

1. Generate Initial Population
2. Evaluate and Rank them based on Fitnesses
3. Produce Population for next Generation
 - Cross Breeds Elite group from Population
 - Mutates some of them
4. Go Back to 3 and evaluate new generation.
5. Repeat Steps 2~4 until the Fitness values reach requirements.

Algorithmic sequence of GA



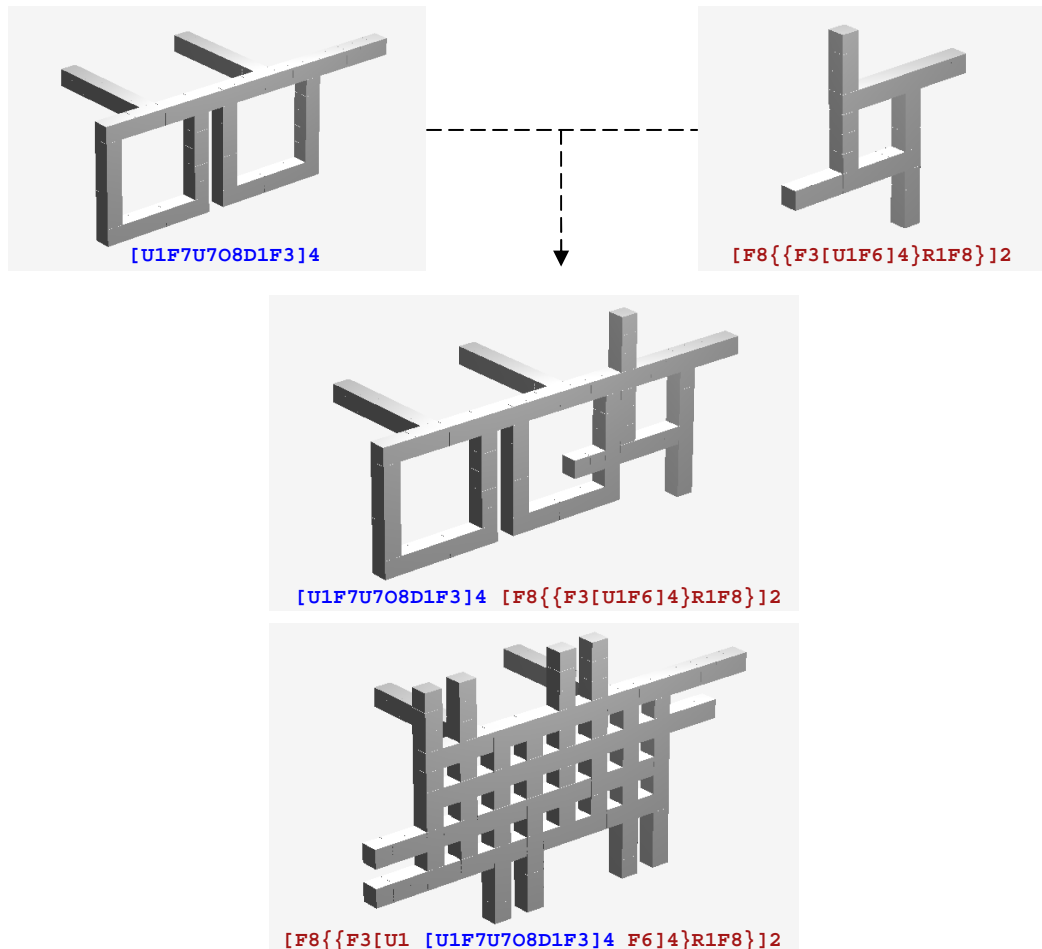
Auto-Generated Design Patterns



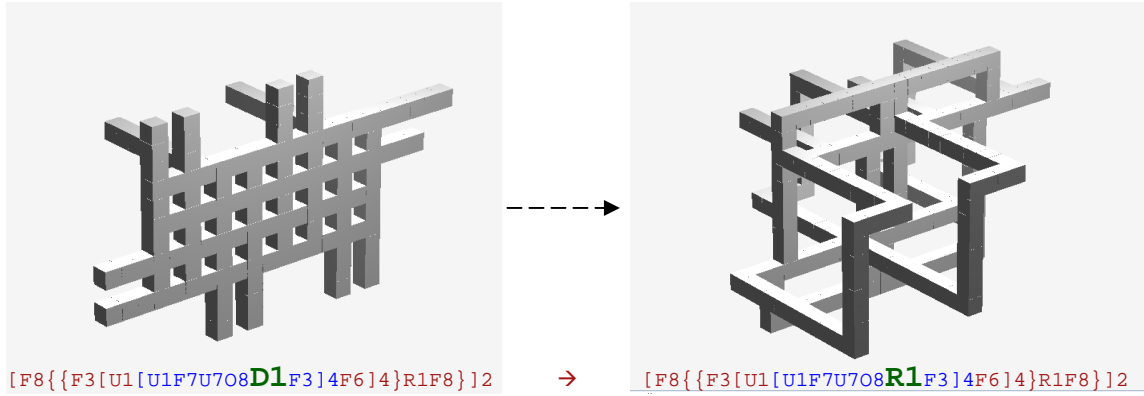
Auto-Generated Design Patterns

4.2.6 Auto-Generation of Design Patterns

Composition of these operators can be automated. By randomly choosing operators and concatenating them into a single instruction for a turtle, one can obtain randomly generated three-dimensional structures. By inserting replicator and push-pop brackets random numbers of times at random locations, structures can have more geometrical variations. This automated process of generation becomes a generator for the program. The initial population is created randomly by this process. The crossover and mutation are two operations that can be applied to instructions for turtles to produce new instructions. The following examples display results of two operations visually in an intuitive manner.



Cross Breeding (Crossover): - Generating New String from Population



Mutation: - Pick random sequence from strings and replace them.

4.2.7 Fitness Functions: Evaluations

The following are various types of evaluations that have been designed and used for this program. Different combinations of the following functions are used in order to direct the growth and selection sequence of structures, each in unique ways. Multiplications of functions are used to combine different characteristics of evaluations, and some weight-factors (ε) are also multiplied in order to implement weights for each function's importance.

$$\text{Fitness Function} := \varepsilon_0 * Eval_0 * \varepsilon_1 * Eval_1 * \dots * \varepsilon_n * Eval_n \quad (0 \leq \varepsilon_i \leq 1)$$

Previously, chapter 3 reviewed three different approaches for integrating multiple results from various fitness functions used in multi-objective optimizations. Those methods are plain aggregation methods, population-based non-Pareto approach, and Pareto-based approach. In this program, I chose to use a simple plain aggregation method. The use of other methods and results from them will be good future explorations. Listed below are various functions that are used for evaluation of architectural structures. Some of their utilities contradict each other.

- EvalAveHeight (Evaluate Average Height)

This function is evaluating every single voxel's height (in the z-axis) and calculating average height. This will select proportionally taller and more slender structures; however, use of this function alone may also select tree-like upside-down structures, flaring upward, which may not always have a good structural performance.

```
foreach (Box t in BoxList)
{
    totalHeight += t.vPos.z;
}
return (float)totalHeight / (float)BoxList.Count;
```

- EvalBoxCount (Evaluate Numbers of Voxels)

This function is counting numbers of voxels in the structure. If you want to grow the structure (or instructions in a string form) to simply use more voxels, this function works.

Normally, this does not impose any morphological tendencies to a certain form.

```
return (float)Math.Pow(T.BoxList.Count, 2.0)
( using power of 2 is also a good strategy to add more importance to a function.)
```

- EvalDenseBound (Evaluate Density relative to a BoundingBox)

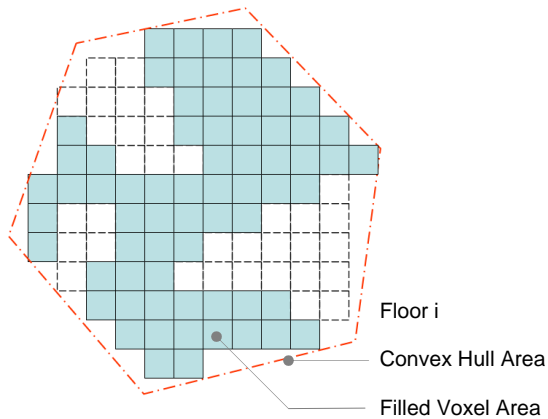
This function calculates density of the structure relative to its bounding box. Bounding Box is a box that encloses an entire structure in x, y, and z directions. This will select more compactly shaped structures and avoid structures with excessive cantilevers, etc. The density relative to Bounding Box is only one way to represent density. The next density calculation using convex-hull may make more conventional sense.

```
return(float)(BoxList.Count/((maxX-minX)*(maxY-minY)*(maxZ-minZ)));
maxX,minX,maxY,minY,maxZ,minZ represent maximum and minimum
values of structure in x,y,z axes.
```

- EvalConvexHull (Evaluate Density based on Convex Hull Area of each floor)

This function calculates the area of convex hull for each floor level in z-axis and finds a ratio of occupied area relative to the convex hull area at the floor.

$$Fitness := \frac{Total\ Area\ of\ Voxels}{\sum Area\ of\ convex\ hull\ (floor_i)}$$



- EvalStress (using Finite Difference Method)

This function revisits the technique, Dynamic Animated Structural Deformation, which I used for my software. This function calculates average amounts of stress at each node in the structure using the finite difference method (Euler method) for a certain period of duration. (This method checks structural deformations per explicit time step and iterates the process.) If the generated structure has more stability and balanced geometry, the stress will be less.

```
Skeleton s = new Skeleton
// calculate Stress at finite time dt=0.001 and iterate
for (int i = 0; i < Duration; i++)
{
    s.calcStress(dt=0.001);
}
// calculate Average Stress at Particle (Node)
float stress = 0;
for (int k = 0; k < numParticle; k++)
{
    stress += (float)((particle[k].stress /
```

```

        particle[k].numStress) / maxStress);
    }
    return stress /= (float)numParticle;

```

This method turns all forces inside the structure into axial forces and calculates the stress. It is a fairly reliable method to find the structural balance of the blocked cluster at a schematic design level; however, the time that it takes to calculate all stresses at every node is too long to use for generations of evolutionary computation.

- EvalCenterMass (Evaluate Center of Mass)

This function calculates the center of mass (assuming the same material density properties throughout a structure) at each floor level and measures the deviation distances of center of mass of each floor level from the center of mass of the entire structure. In other words, if the center of mass location at each floor fluctuates less, your structure is considered to have more balance. This is relatively unsophisticated compared to the previous methods using finite difference method, but the time for calculations is faster.

- EvalExposedFace (Evaluate numbers of exposed open faces)

This function calculates numbers of open faces exposed to the air rather than blocked by their neighbors. This is useful when we want to know roughly how much open window area we want to propose for the new structure.

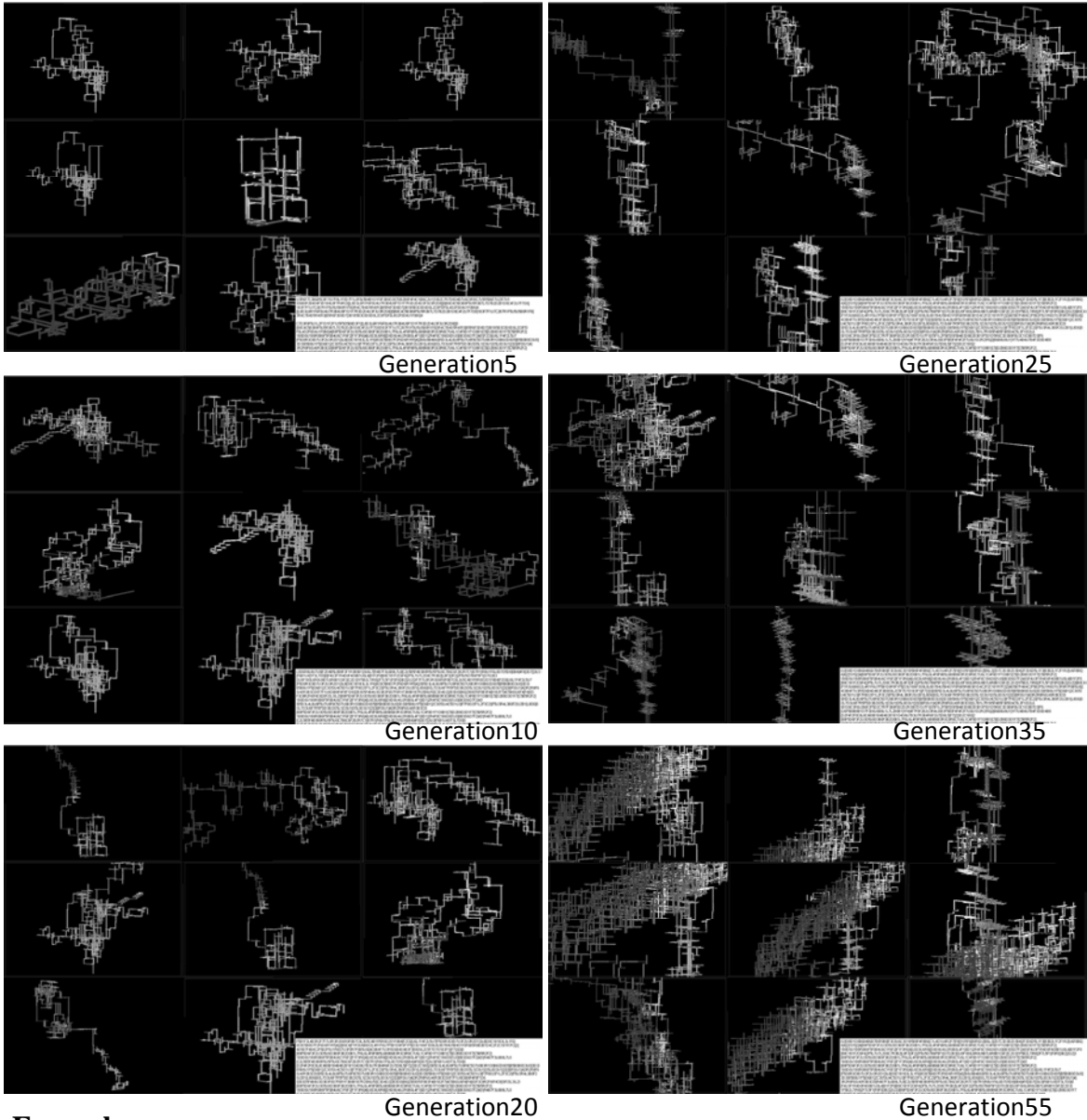
```

//Total maximum 6 open windows per voxel
foreach (Box t in BoxList)
{
    totalOpenFaces += t.OpenWindows;
}
return (float)totalOpenFaces / ((float)BoxList.Count * 6);

```

4.2.8 Evolutionary Runs

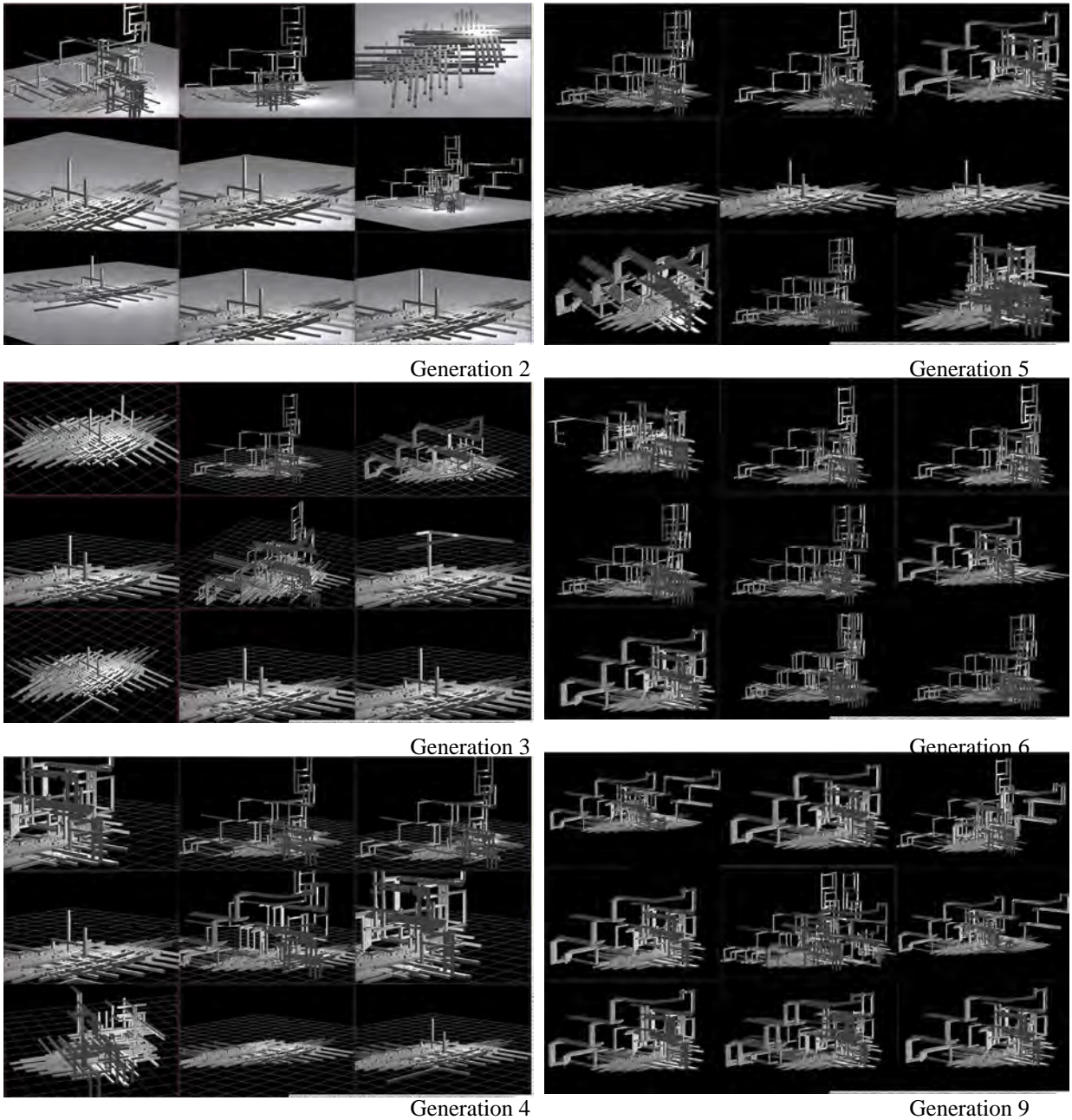
Initial populations are randomly produced. I have listed screenshots of several results from different combinations of fitness functions in the following pages.



Example:

```
Population = 20;
Fitness = EvalBoxCount() * EvalAveHeight() * EvalDenseBound();
```

Example of simple evaluation up to 10 generations is calculated. No structural evaluation is implemented. Initial populations are too small, with the consequence that the later populations are highly biased by them.

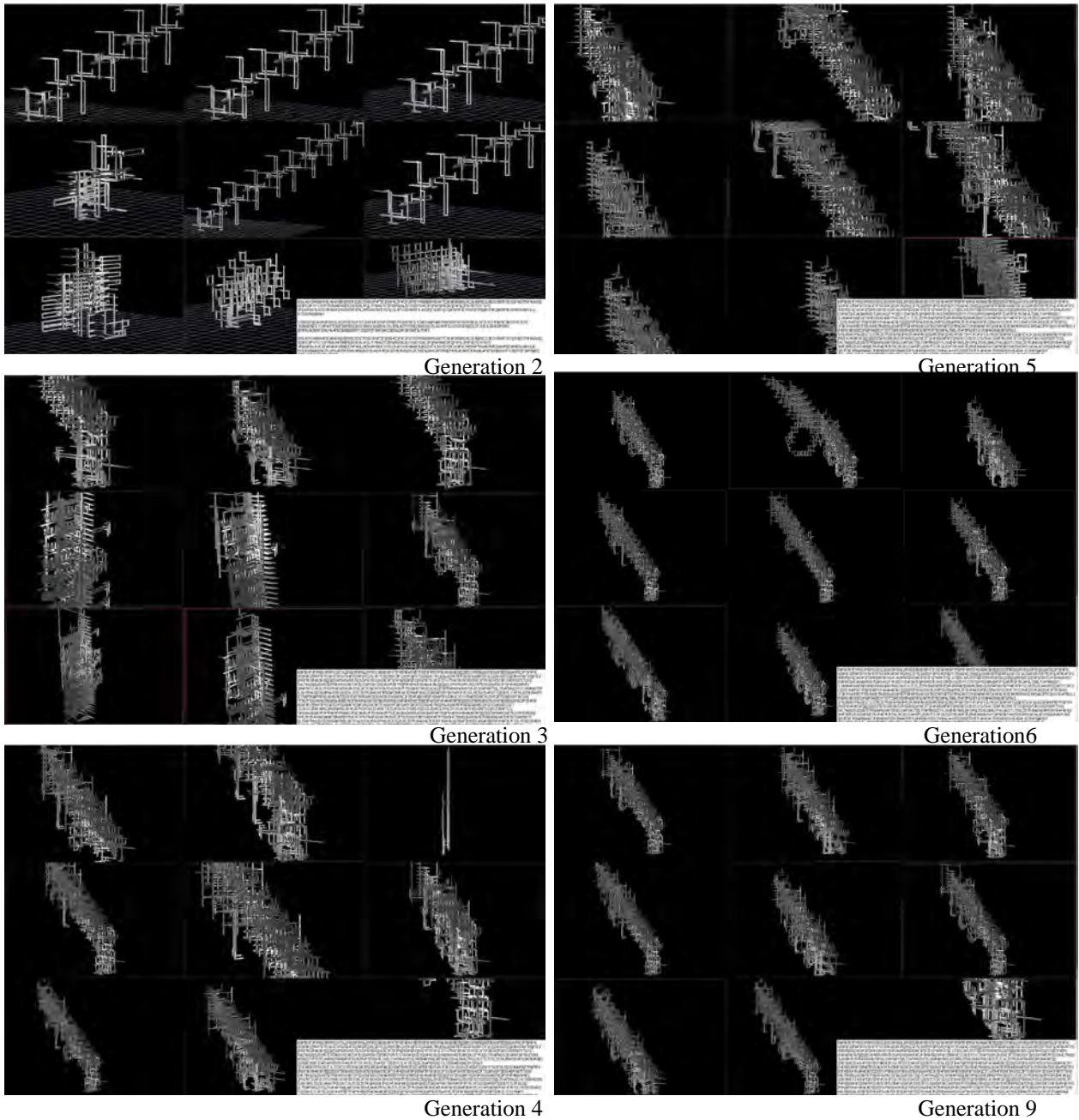
**Example:**

```

Population = 50;
Fitness = EvalBoxCount() * EvalConvexHull();

```

In this case, the fitness evaluation tends to give higher values for flatter, more balanced, and larger (with more voxels) structures. In generations beyond 10, high-scoring schemes from earlier generations became dominant species and the search process started to be stagnated at local maxima by producing similar structures. Higher mutation rates and larger populations need to be considered to get improvement.

**Example:**

Population = 50;

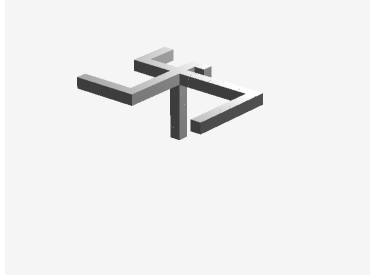
Fitness = *EvalBoxCount*() * *EvalAveHeight*() * *EvalDenseBound*();

In this case, the fitness evaluation tends to select taller and denser structures; however, there are not enough evaluations relating to structural efficiencies. So, the generative process produced leaning towers. I found that multiplying **EvalStress** using Finite Difference Method would solve this problem, but the calculation time was not practically useful relative to an accessible current computational speed. Here again, biases from earlier generations are too strong, producing apparent stagnation again at the local maxima.

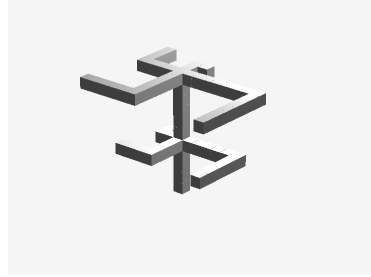
4.2.9 Procedural Representation Using Parametric L-System

Procedural (functional) representation allows having more concise instructions for form generations. Parameters n and m inside the Expression-operator are used to parameterize the growth of forms and they are executed based on conditional statements. Explanations of Expression-operators are as follows.

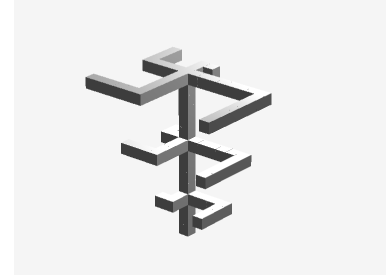
Expr1(n, m): If $(n > 1) \& (m > 0)$ \rightarrow $\{ \{ F(n*2)R1F(n*2) \} R1 \text{ Expr1}(n-1, m) \}$
 Else if $(n > 0) \& (m > 0)$ \rightarrow $U1F(3+m)D1 \text{ Expr1}(n+m-1, m-1)$



Expr1(4,1)



Expr1(4,2)



Expr1(4,3)

Expr1($n=4, m=3$):

$n=4 \ m=3$

Expr1(4, 3) = $\{ \{ F8R1F8 \} R1 \{ F6R1F6 \} R1 \{ F4R1F4 \} R1 \{ F2R1F2 \} R1 \} U1F6D1 \text{ Expr1}(3, 2)$

$n=3 \ m=2$

Expr1(3, 2) = $\{ \{ F6R1F6 \} R1 \{ F4R1F4 \} R1 \{ F2R1F2 \} R1 \} U1F6D1 \text{ Expr1}(2, 1)$

$n=2 \ m=1$

Expr1(2, 1) = $\{ \{ F4R1F4 \} R1 \{ F2R1F2 \} R1 \} U1F6D1 \text{ Expr1}(1, 0)$

$n=1 \ m=0$

Expr1(1, 0) = Terminates Operation ($m \leq 0$)

Expr1(4, 3) :=

$\{ F8R1F8 \} R1 \{ F6R1F6 \} R1 \{ F4R1F4 \} R1 \{ F2R1F2 \} R1 \} U1F6D1 \{ \{ F6R1F6 \} R1 \{ F4R1F4 \} R1 \{ F2R1F2 \} R1 \} U1F6D1 \{ \{ F4R1F4 \} R1 \{ F2R1F2 \} R1 \} U1F6D1$

Mutation of Parameters

Current Scheme = {Expr1, Expr2, ..., ExprN};

(Example Scheme)

Expr1(n, m) = if($n > 3$) then Expr2(m)Expr7($n-1$)
 If($n > 0$) then Expr6($m-1$)

(Example Expressions)

If($n > 0$) then Expr6($m-1$)

\rightarrow

If($n > 3$) then Expr4($m+1$)

(Mutation example)

Rule strings for form can be composed based on several sets of Expressions with various parameters, and they can cross-reference each other based on varying conditional statements. Concise forms of expressions can be mutated and recombined to create newer rules to search for better fitness. Parameters can be directly mutated, and more pinpointed updates are possible with these processes.

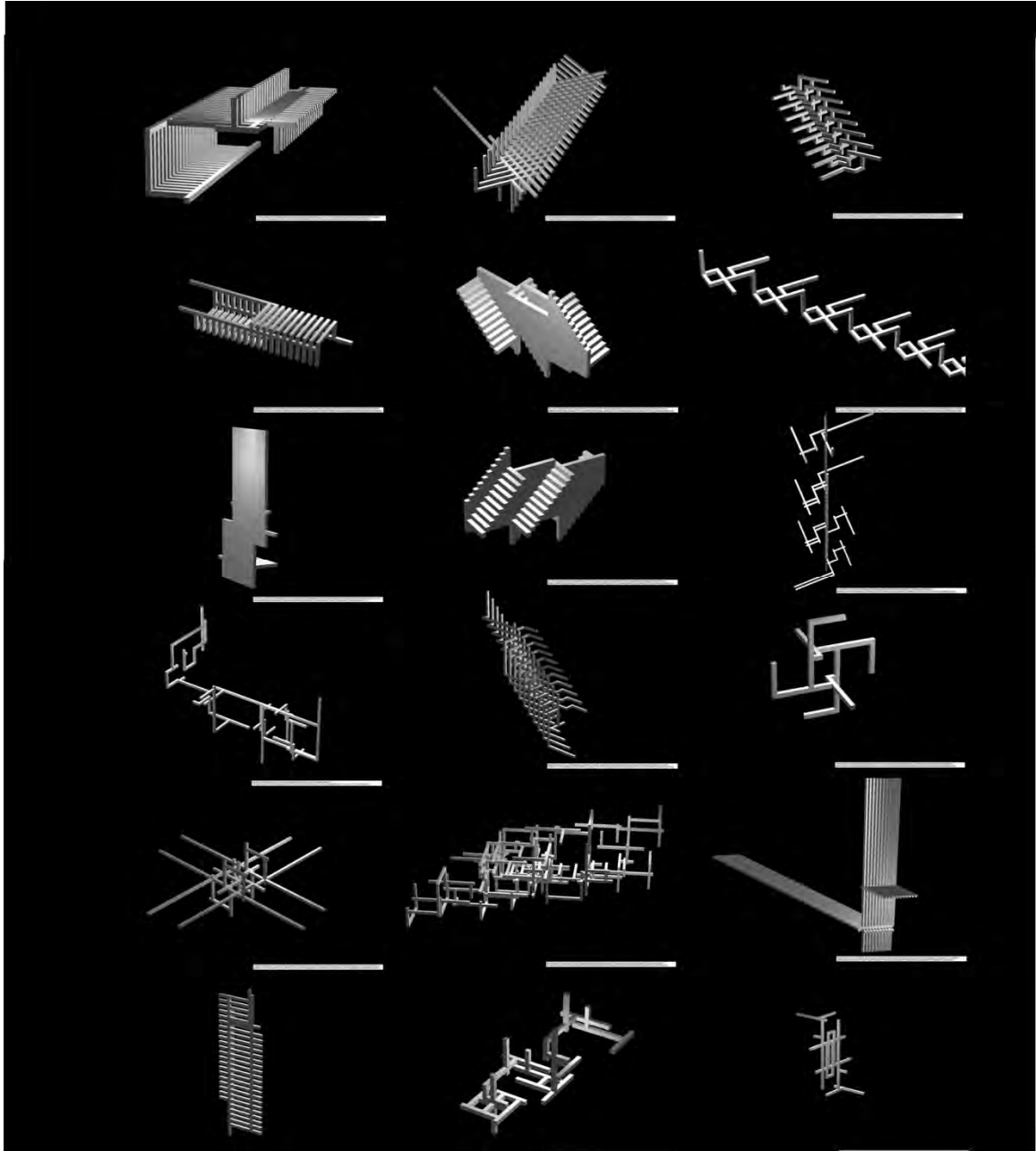


Figure 4.3. – Design Patterns created by Parametric Representations

4.2.10 Future Work

Parametric representation will create an algorithmic system that is similar to genetic programming (GP), and the potential of evolutionary computation using parametric representation needs to be further investigated. Differences in cross-breeding and mutation operations between non-parametric and parametric functional representations will be a particularly important investigation. The system proposed in this section can implement more specific architectural constraints based on various fitness functions. More specific architectural information such as occupancies, site configurations, far-area-ratio (FAR), and so on can be implemented for more practical applications.

Some of the functions are computationally too time-consuming. The EvalStress function is suited for visualizing structural performance; however, some other simpler and faster calculation method is anticipated for this type of generative algorithmic framework that deals with numbers of populations of schemes.

User interaction can be easily implemented by having users select particular schemes that are preferable based on their visual observations. Regardless of the values from fitness evaluation, some scheme may become important for a certain user due to various reasons. This is the interactive GA framework that I reviewed in the last chapter. This functionality will shed light on some characteristics that are difficult to define as fitness functions. Since all schemes can be represented by (essentially) a series of strings, even if the characteristics are not mathematically formulated, those characters are already encoded inside of the abstract form of strings.

Chapter 5

Experiments in Growth and Adaptation

Introduction

In this chapter, implementations of the third category of the computational methods from chapter 3 – growth and adaptation – are introduced through two examples of design problem frameworks. The first example shows growth models using diffusion-limited aggregation (DLA), and the second example shows physical implementation in architecture using a multi-dimensional optimization method called the Nelder-Mead method. The concept of growth and adaptation is the main focus of the thesis, and conceptual foundations introduced in this chapter from these experiments will lead to the more elaborate applications in the next chapter.

5.1 Growth: Diffusion-limited Aggregation (DLA)

In this section, diffusion-limited aggregation (DLA) will be focused on as a method to realize the third category of computational implementation – growth. Diffusion-limited aggregation is a process of accretion over time and is observed in many systems such as electrodeposition, mineral deposits, snowflake formations, dielectric breakdown (lightning path), and even in living organisms, such as the growth pattern of coral. Witten and Sander (1981) proposed a theory that explains DLA growth, and their idea is that randomly walking particles launched from distant points stick to the surface of the growing cluster when they arrive at a site adjacent to the aggregate.

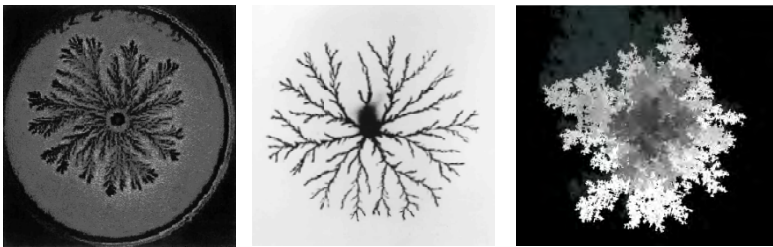


Figure 5.1.1 – DLA Examples: Colony of *Paenibacillus dendritiformis* bacteria (left) Zinc electrodeposit (middle) from (Sander, 2000), Plan view of model by the author (right).

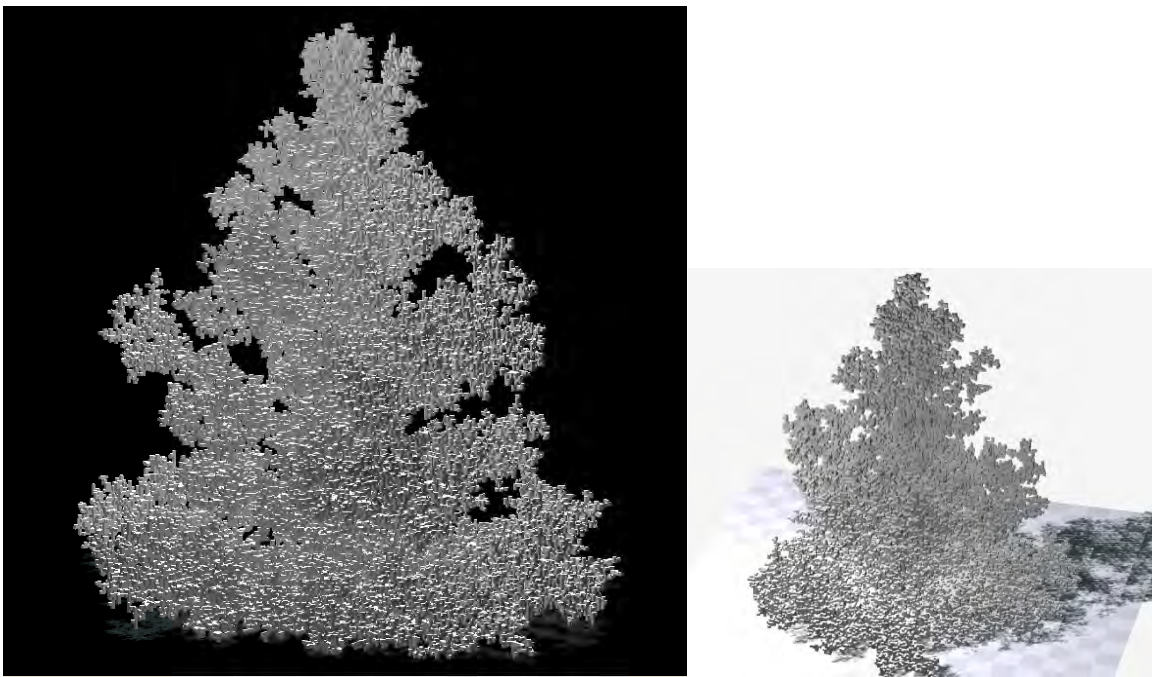


Figure 5.1.2 – Perspectives of DLA model with 10,000 cells. Models by the author.

5.1.1 Laplacian DLA Model Based on Probability

The DLA-cluster can also be interpreted as an aggregate where the formation is controlled by the probability of particles reaching the cluster. Witten and Sander provided the mathematical explanation that the random-walk DLA model can be rewritten as the Laplace form of the diffusion equation based on the probability of a particle at position x at time t as $u(x, t)$. (Witten and Sander, 1981).

The change of u over time Δt is given by

$$\begin{aligned} u(\vec{x}, t + \Delta t) - u(\vec{x}, t) &= \frac{1}{2d} \sum_{i=0}^{2d-1} [u(\vec{x} + \Delta\vec{x}_i, t) - u(\vec{x}, t)] \\ &= \frac{1}{2d} \sum_{i=0}^{2d-1} [u(\vec{x} + \Delta\vec{x}_i, t) - 2u(\vec{x}, t) - u(\vec{x} - \Delta\vec{x}_i, t)] \end{aligned}$$

, where d is the dimension of the lattice. The above can also be written as,

$$u(\vec{x}, t + \Delta t) - u(\vec{x}, t) = \frac{\partial u}{\partial t} * \Delta t$$

From the above two equations, they can be rewritten as

$$\frac{1}{2d} \sum_{i=0}^{2d-1} [u(\vec{x} + \Delta\vec{x}_i, t) - 2u(\vec{x}, t) - u(\vec{x} - \Delta\vec{x}_i, t)] = \frac{1}{2d} \sum_{i=0}^{2d-1} \frac{\partial^2 u}{\partial x_i^2} (\Delta x_i)^2 \quad (1)$$

, which generalizes to the Laplace form of diffusion equation,

$$\frac{\partial u}{\partial t} = \eta \nabla^2 u$$

, where η is the diffusion constant.

Solving the equation (1) with the boundary conditions $u(x, t)=0$ where x is inside the DLA cluster and $u(x, t)=\text{constant}$ where x is an infinite distance away from the center will provide the probabilistic growth model of DLA, and we can gain the same results from random walk DLA (Witten, 1981).

Theoretically, this probability for potential growth area can be assigned based on more complex information relating to architectural constraints such as light, view, circulation, structure, and so on.

This experiment was started by measuring views and distances among the unit cells within 10 units' distance around the potential next deposition locations in order to maintain a certain level of privacy among the units. As a result, growth of thin diagonal branches was observed and they all kept a certain distance from others. By biasing the probability in south-north orientation, the growth toward a certain orientation can be directed based on a specific condition of solar radiation. The growth can be constrained within a bounded zone as well. In this case, instead of branches, formations of parallel layers of strata were observed.

In addition to aggregation, reductive processes can be used by introducing a predator that eats the units which have lower values among the existing cells. This process will maintain a better overall value for the cluster by replacing old cells with new ones as if it were metabolizing them.

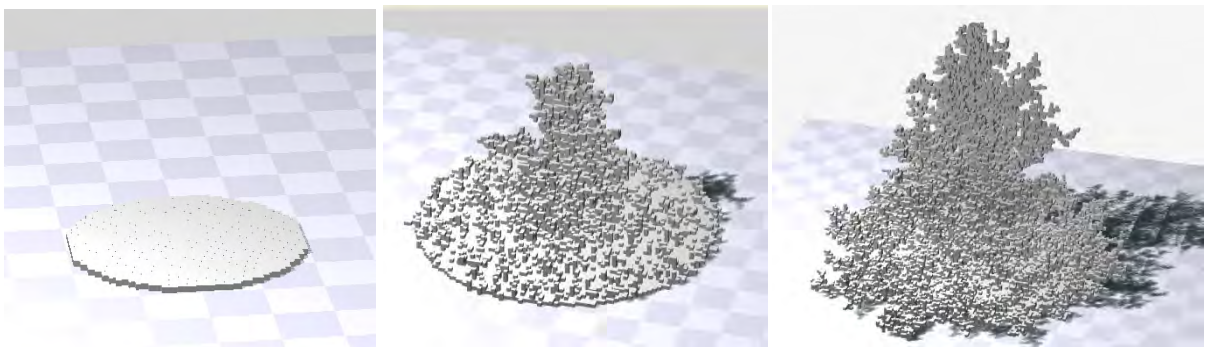


Figure 5.1.3 – Initial Seed as a planar surface

5.1.2 Architectural Applications of DLA

1) Random walk DLA model:

Firstly, provide the cell at the center of the lattice as a seed point. Pick a square at the perimeter of the lattice and place a randomly walking particle on that square. With each time step, this particle moves to one of the adjacent squares, top, down, left, right, above, or below. The particle continues to move until it arrives at the empty cells adjacent to any existing cells that are already occupied, and it sticks there. Release a next particle and repeat the above process. The figures below show results of structures composed of 50,000 and 100,000 particles. In this experiment, an initial seed cell is located at $(0, 0, 0)$, and particles move within the positive area of z-axis. The plan view below resembles typical DLA patterns in two dimensions.

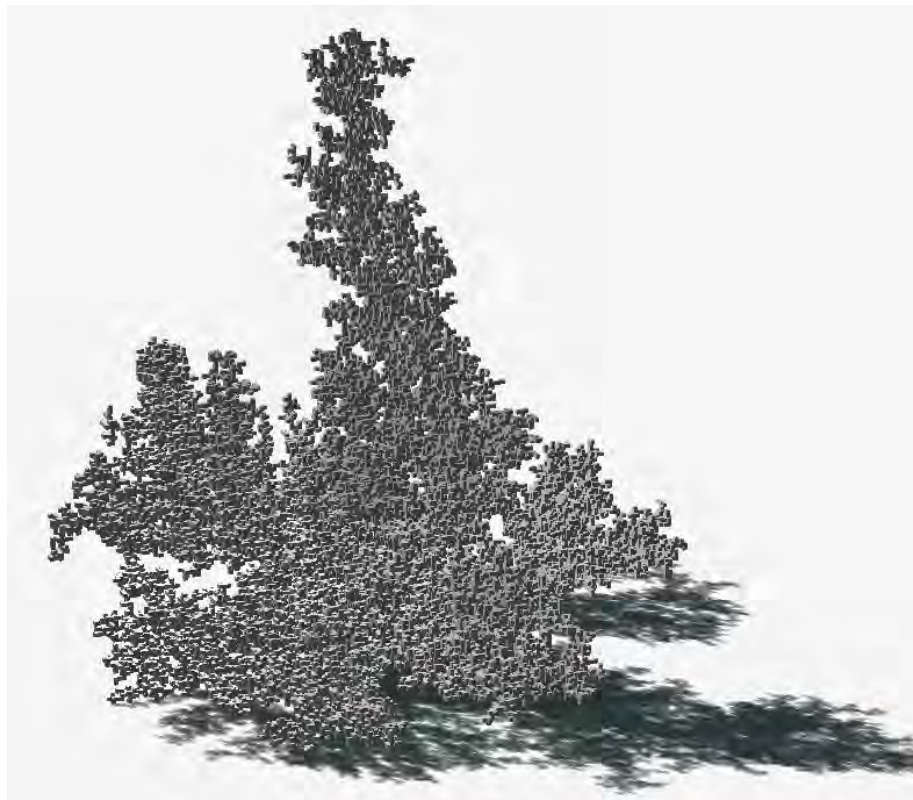


Figure 5.1.4 – Three-dimensional clusters created by Diffusion-Limited Aggregation. DLA cluster with 10,000 cells. Approximately 550% more opening areas than the structure using simple 10x10x100 tower configuration.

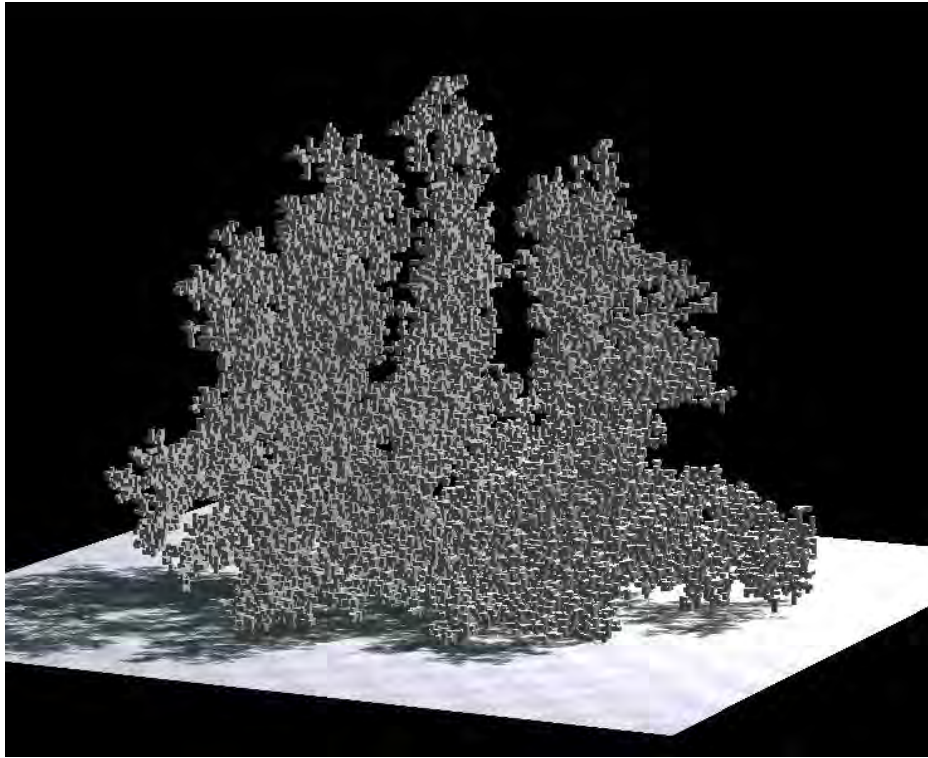


Figure 5.1.5 – Three-dimensional clusters created by Diffusion-Limited Aggregation. Examples of clusters by various probabilities for potential growth area.

2) Probability based on Density/Openness:

Getting an idea from this probability-based DLA model, probability for each square in the lattice is assigned based on a certain logic. In architecture, when we design the layout of units, especially for residential buildings, natural lighting from all sides is a critical issue. It is not desirable to have two units facing each other in too close proximity. Firstly, the probability is calculated based on the numbers of units around the potential unoccupied cells within a certain distance (5 and 10 units' length). This is the equivalent of calculating a density around a certain radius of bounded area. The unoccupied cell with higher density or openness to surrounding conditions will have less probability to be occupied by the next time step. By using the density measure for a larger area (10 units' length rather than 5 for the area by which to measure density), longer branches are gained. Hence, for the structure as a whole, the average numbers of open faces per cell become

higher, and the resulting structure has more exposure to daylight and views compared to a simple extrusion of rectangular plan configurations with the same total volume. An average performance of daylight exposure over a certain duration of growth periods is a particular interest of this project. Solutions at each time step may not provide the optimal solution for specific conditions at the step; however, the DLA method continues to search for good configurations throughout the growth processes over time.

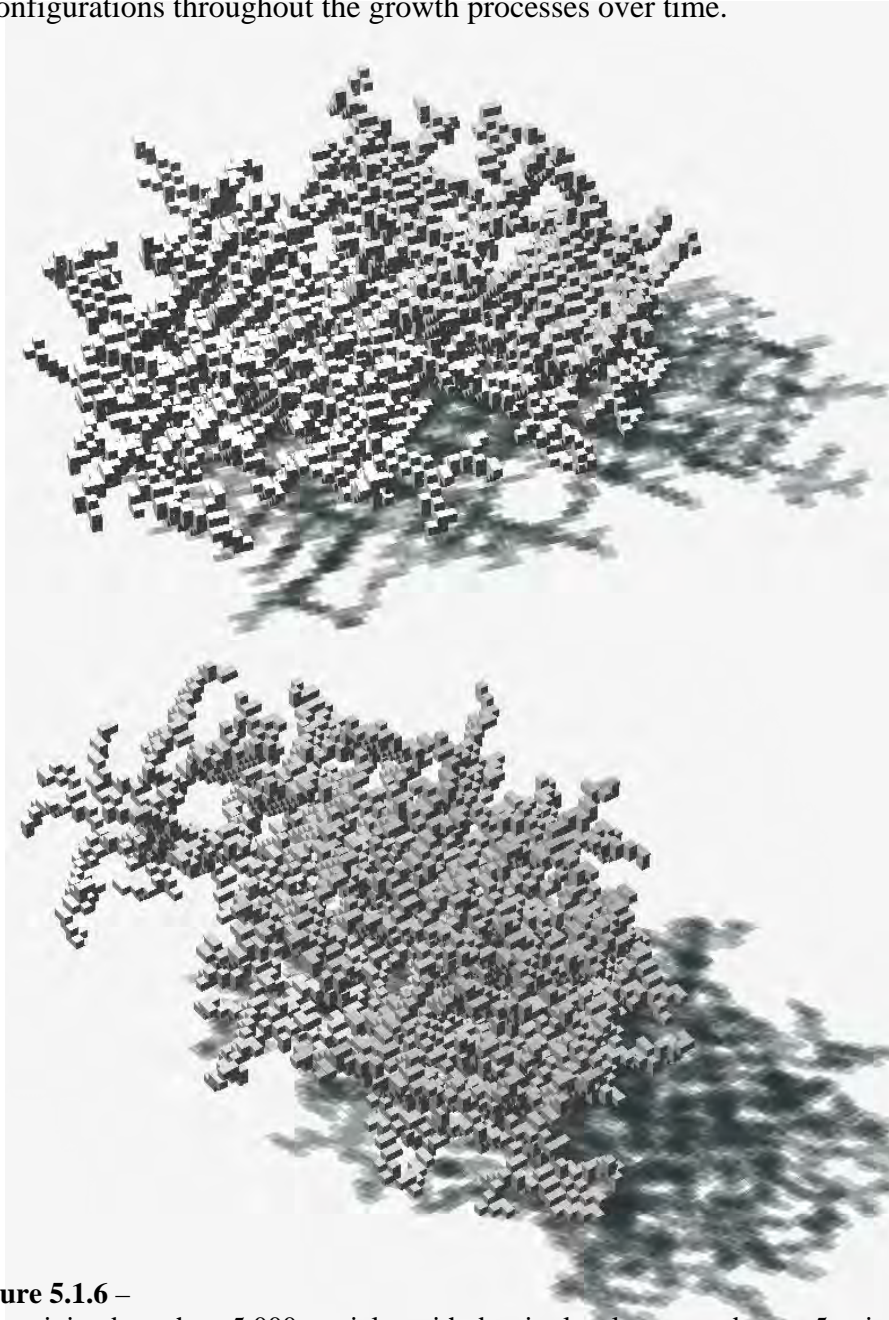
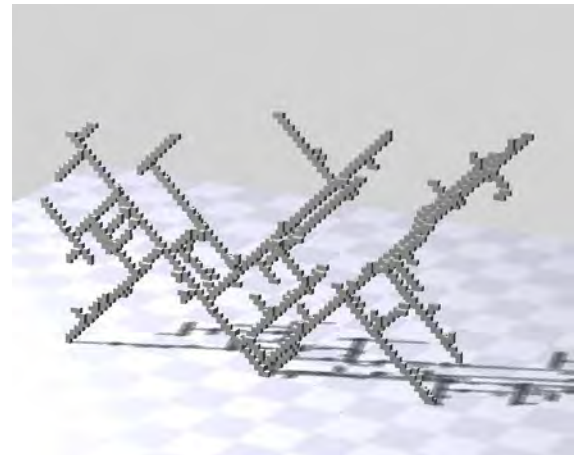
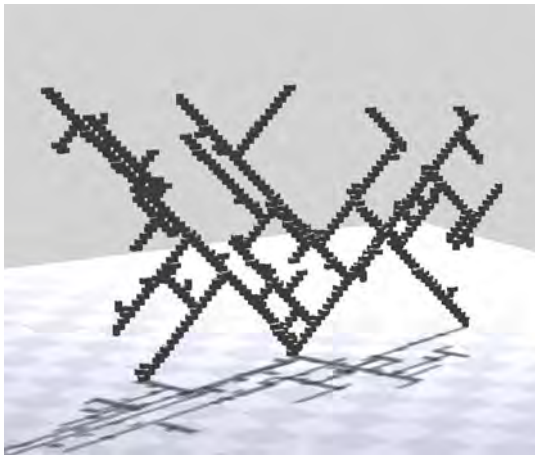


Figure 5.1.6 –
Intertwining branches: 5,000 particles with density level measured up to 5 units' distance

3) Bias Toward the North-south Direction:

Adding bias toward the north-south orientation by increasing the values for calculating density on that direction enhances or restricts the growth along that direction. In architecture, the lighting from the south is more desirable for the northern hemisphere of Earth, and I intended to break the symmetry of configurations in the north-south direction. Since there is no boundary limit in the lattice, the diagonal branching extends while maintaining the maximum distance for density calculation. The bias suppressing the growth in a two-dimensional plane produces a typical two-dimensional DLA pattern, and there is a similarity to the density distribution pattern of many urban areas. This is also stated in a similar study (Batty, 2005).



Applied Bias at the N-S direction:

```
for (int i = -10; i < 11; i++)
{
  if(i!=0 && ii+i>0 && ii+i<gx)
    val += 2 * grid[ii + i][jj][kk];

  if(i!=0 && jj+i>0 && jj+i<gy)
    val += grid[ii][jj + i][kk];

  if(i!=0 && kk+i>0 && kk+i<gz)
    val += grid[ii][jj][kk + i];
}
```

Figure 5.1.7 – Controlled bias toward North-South direction

I kept the implementations of the probability rules at an abstract level in this experiment. However, more specific information can be described. This probability for each square location can be based on more architectural evaluation of the spaces, such as sunlight exposure or accessible pedestrian circulation. The coral growth model by J. Kaandorp (1994) uses nutrient concentration level around the developing structure as the probability for the DLA while using a fluid dynamics model to simulate nutrient movement. In the case of architectural applications, the flows of nutrient can be replaced by the flows of occupants, pedestrians, car traffic, and so on. This is also a concept relevant to many topology optimization techniques. Shapes of automobiles can be sculpted by the movements of particles that simulate the aerodynamic flow of air. This reverse engineering process can be applied to architectural form, making use of the behaviors of occupants as an active sculptor of the space.

4) Constrained Boundary Condition for Growth and Swapping

Most of the developments in building scale are constrained within a certain bounded area in plan view (an xy-plane) and a height. It is more likely to be meaningful to find out the growth within a constrained bounded region. The result using 50,000 particles within a 40x40x40 grid area with a 10-maximum-unit-distance for the density measure shows a series of parallel strata diagonally occupying the grid area instead of forming one-dimensional linear branches.

Introducing a predator that consumes the units which have lower values, based on the density calculation, is another idea I used. The unit with too-high density is considered to be useless without having enough exposure to daylight. Hence, it is eliminated. Instead, I

reproduced a new cell at the location with highest potential daylight exposure value. Repeating this process, “swapping,” is expected to improve the overall value for the cluster of units. I observed gradual extension of branches and reduction of densely populated areas as seen in the figure below. Applying the same process 2,000 times for the above cluster in the constrained boundary, I recognized clear emergence of strata.

The basic DLA process is primarily an accretive process, and the introduction of an elimination process can be interpreted as an implementation of selection among the cells.

- Constrained Boundary Conditions

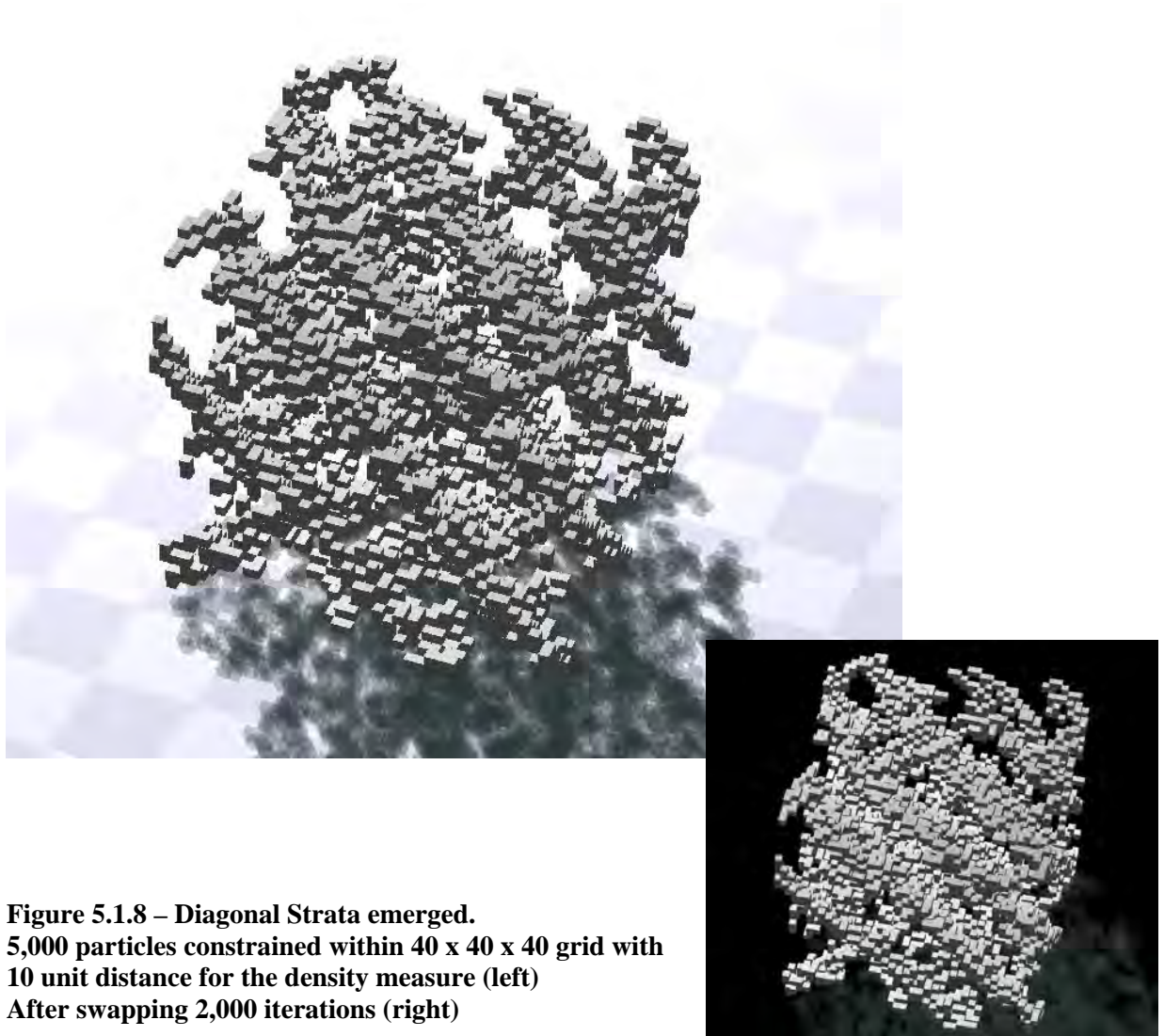


Figure 5.1.8 – Diagonal Strata emerged.
5,000 particles constrained within 40 x 40 x 40 grid with
10 unit distance for the density measure (left)
After swapping 2,000 iterations (right)

- Swapping Process

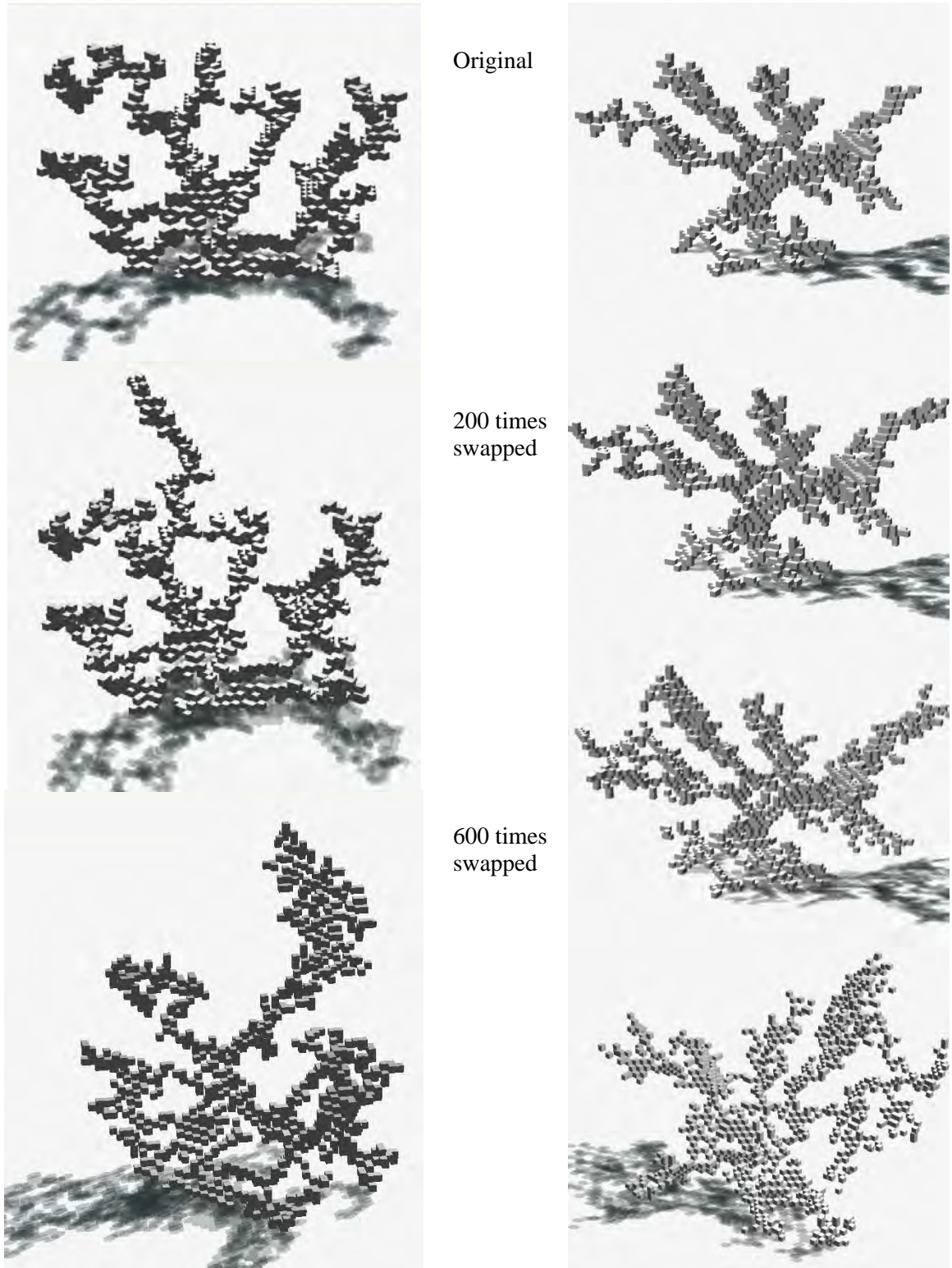


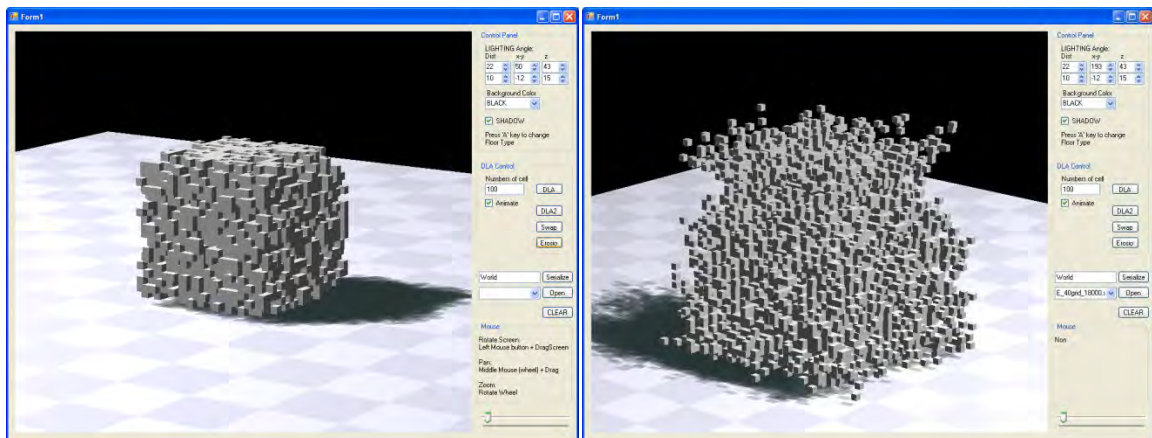
Figure 5.1.9 – 1,000 particles with 10-unit distance for the density measure and swapped 600 times. Branches extend outward after the swapping.

5) Erosion (Subtraction Algorithm):

The last method, erosion, is the inverse process of accretive DLA process by elimination. The erosion process repeats the following: The process starts from the solid lattice filled with cells, and releases a predator particle inside a field of forces from a randomly assigned point at some greater distance. The particle flows along the force field defined by various vector field functions and scrapes the cluster when it hits the solid parts. The parts where more particles flow in will be eaten away faster. For the figures below, the following torus knot equation is used to increment the flying “eliminator” particle’s trajectories.

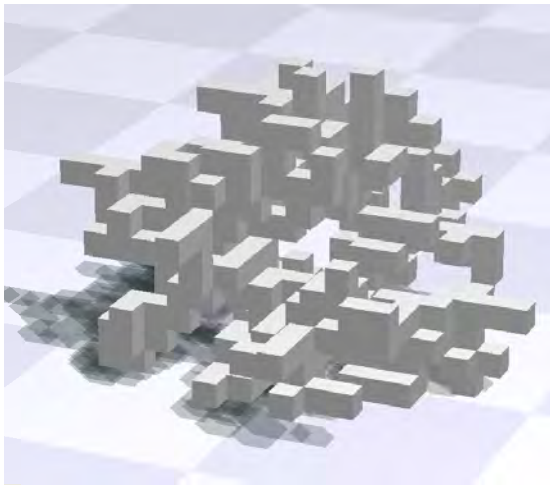
$$\begin{aligned}x(t+\Delta t) &= x(t) + 30 * (2 + \cos(q * \phi / p)) * \cos(\phi); \\y(t+\Delta t) &= y(t) + 30 * (2 + \cos(q * \phi / p)) * \sin(\phi); \\z(t+\Delta t) &= z(t) + 30 * (\sin(q * \phi / p)); \\p &= 4, \quad q = 5;\end{aligned}$$

This study required further investigation to gain more visual clarity, and the choice of the proper vector fields also needs to be studied more. This technique can be used for simulating soil erosion by river currents and other similar phenomena.

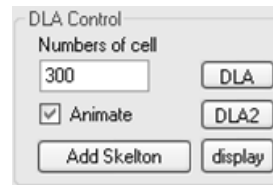


Figures 5.1.10 – Erosion Process

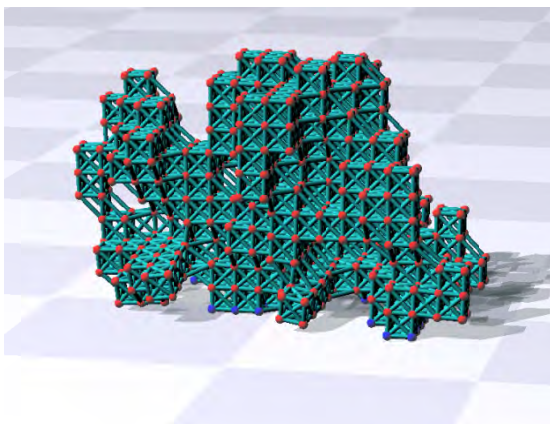
6) Animated Dynamic Structural Deformation



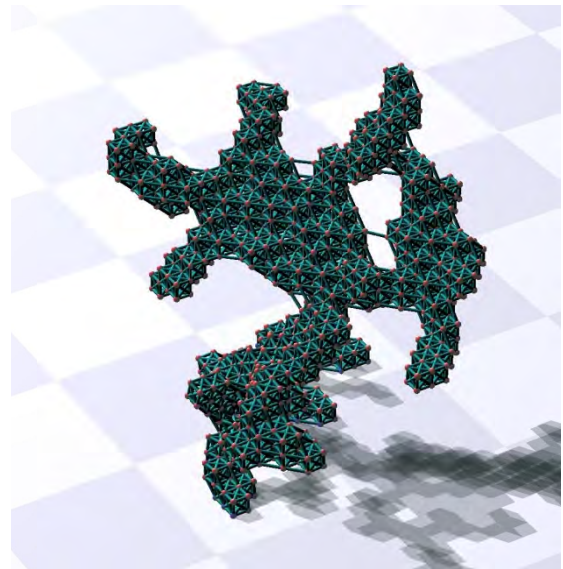
- Normal Display



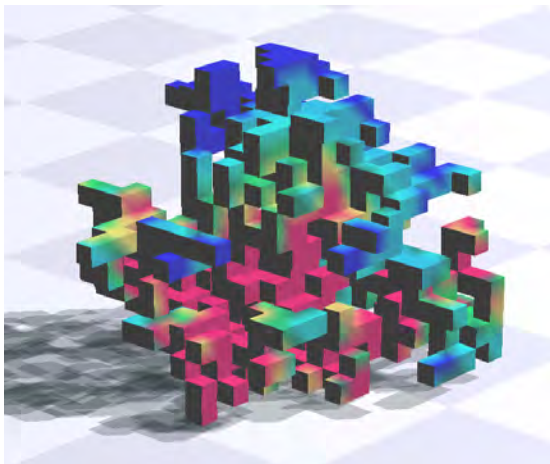
Software allows three different types of display modes. Massing schemes generated by DLA logic will be replaced by 3-D space frame structures in order to approximate structural balances based on all axial forces. This functionality is intended to provide users an intuitive understanding of structural behaviors at the schematic level in a real-time manner.



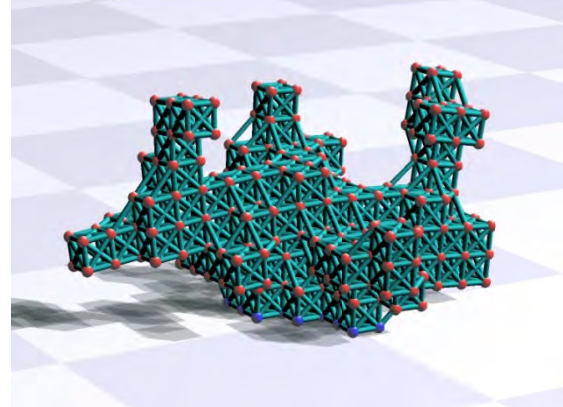
- Truss Display with Shear Bracing Members



Design Example from DLA Studies (Auto generated form)



- Animated Structural Deformation with Stress



Design Example from DLA Studies

Figure 5.1.11 – Animated Dynamic Deformations

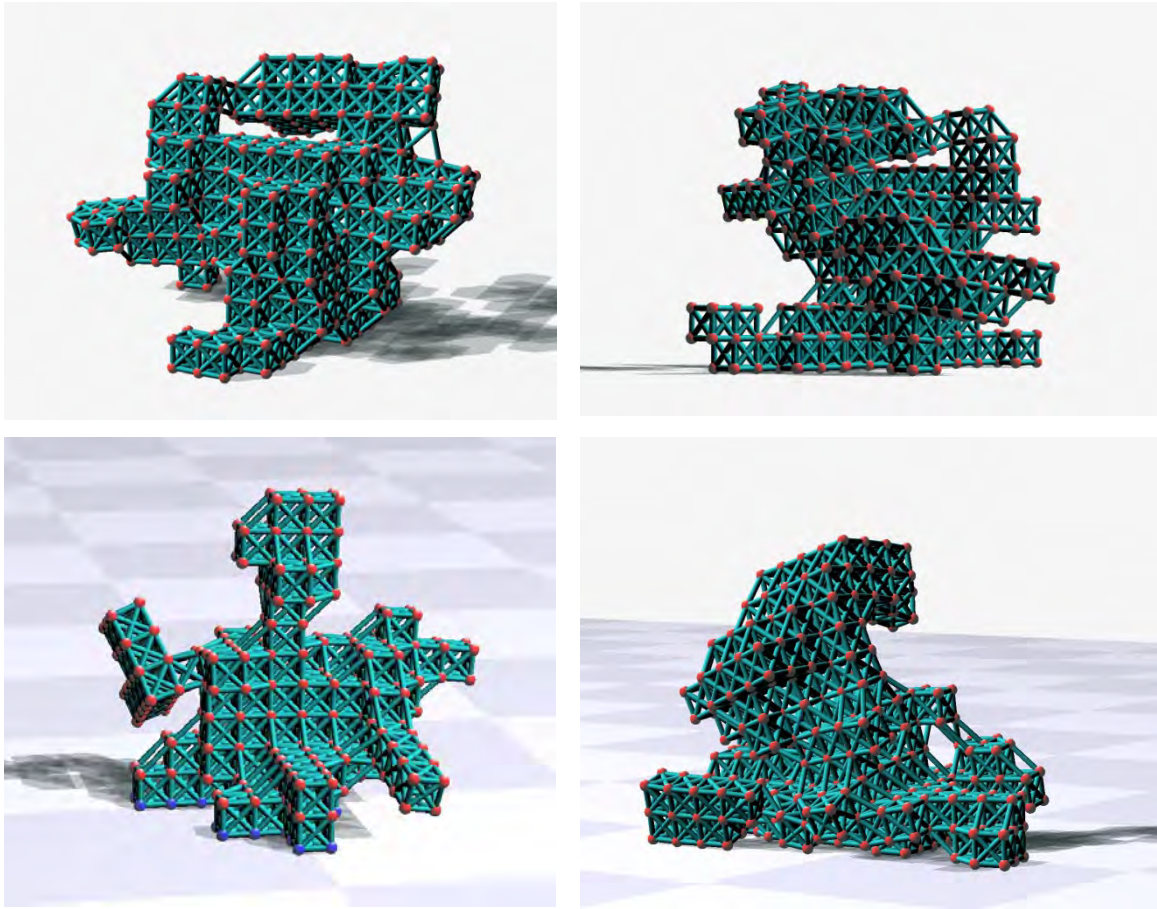


Figure 5.1.12a – Design Examples from DLA Studies (Auto Generated Forms)

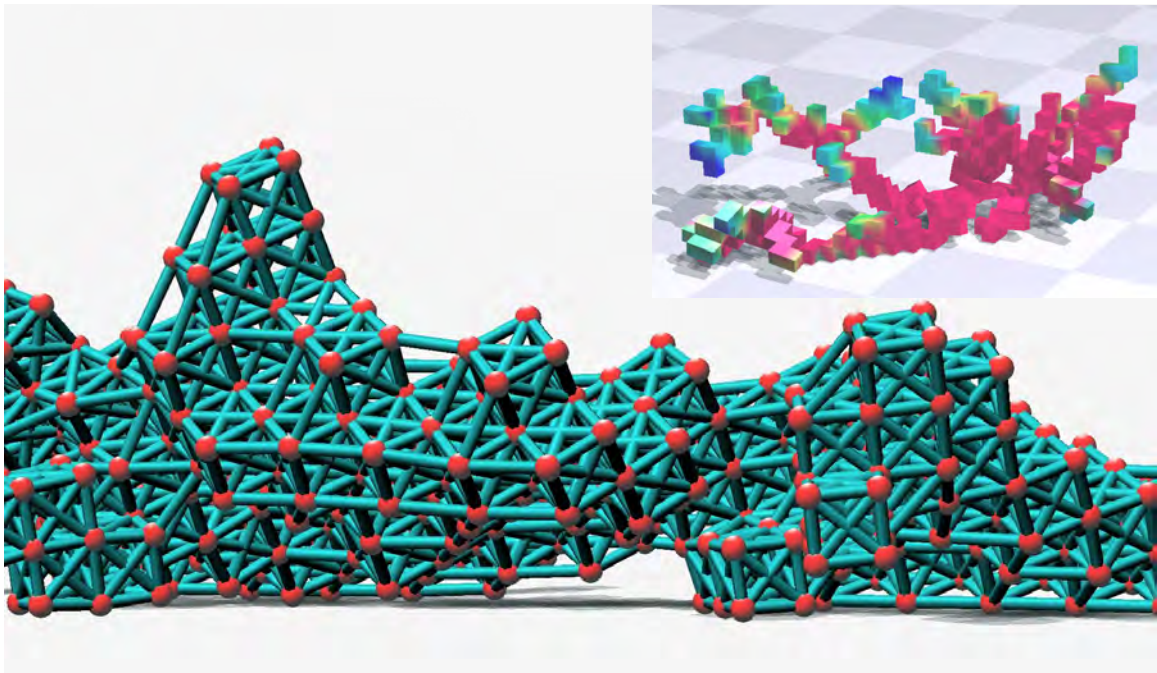


Figure 5.1.12b – Unsuccessful implementation example: Failing Structure

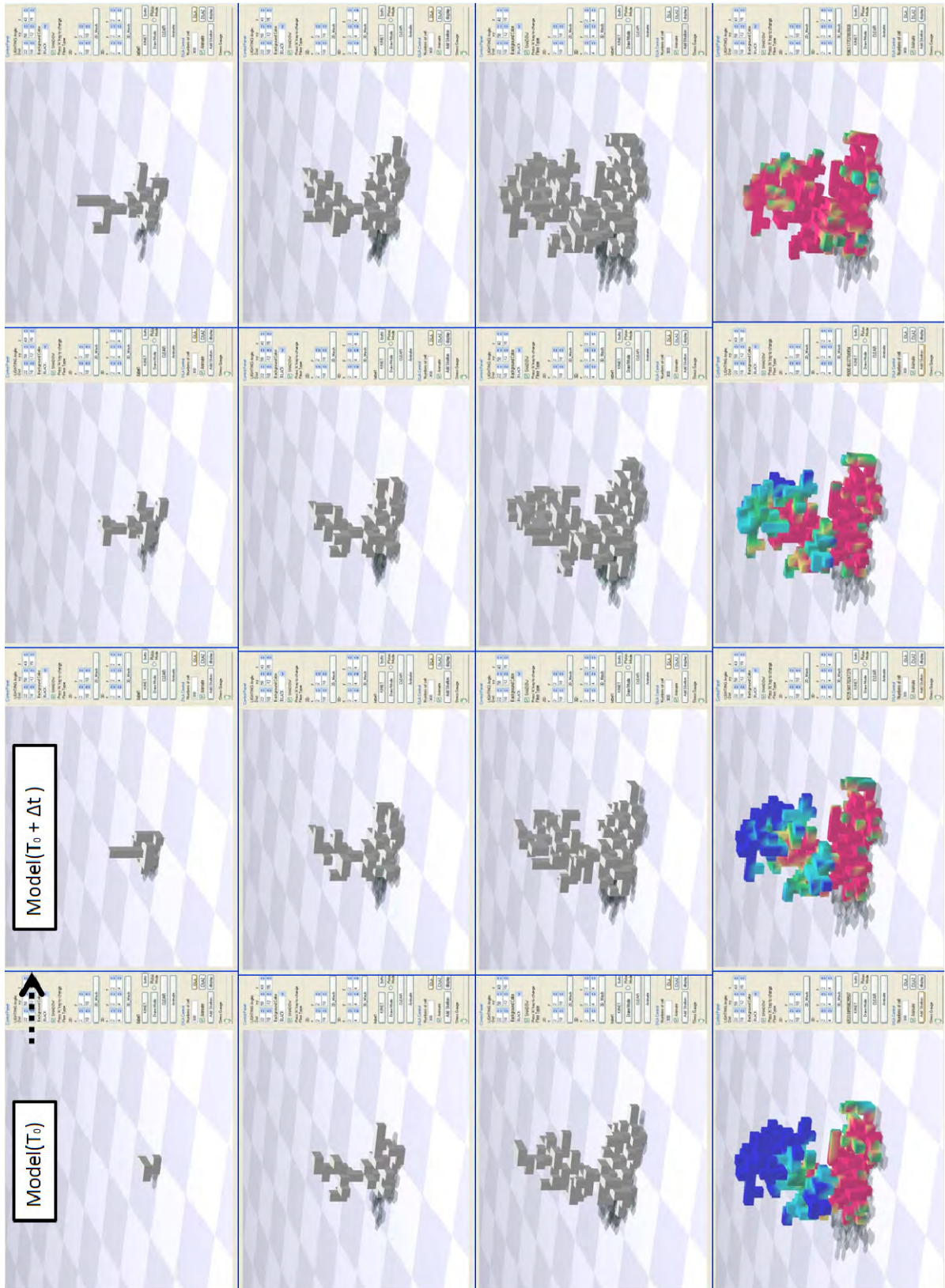


Figure 5.1.13 – DLA Growth Sequence and Animated Dynamic Structural Deformation

5.1.3 Conclusion Regarding DLA Experiments

In this section, I listed several growth models developed from a simple DLA model application. A DLA model is probably not the most reliable method to derive the optimal configurations for specific conditions at a specific moment. However, its ability to stimulate growth toward specific characteristics from simple probability-based stochastic selections is robust and suited for describing growth patterns of spatiotemporal structures. If the search space for the problem is relatively small, the conventional design strategy can still analytically derive more deterministic and reliable solutions. What is more intriguing about this process is that the system can provide quite robust solutions for constant and endless changes, and the system is designed to maintain its balance as a whole. For instance, accretive growth patterns of corals are simulated by a DLA model and are known to closely follow its algorithmic pattern (Kaandorp, 1994; Merks, 2003). Their growth behaviors appear to be quite transient during their development, though their systems of growth are known to maintain effective forms to absorb nutrients in the fluid.

In principle, beyond physical/environmental constraints, programmatic/social issues relating to occupancy types, social issues, architectural programs, and code/zoning constraints can be implemented by assigning proper probability fields over the site under consideration. The probability-based DLA model has a remarkable resemblance to the construction sequences of housing developments such as Kowloon, where people build additions in the most probable and appealing locations at each time step. This seemingly momentary and transient attitude toward construction is definitely not proper for the

design of a single residence with limited site area. However, here is a question: “What if the system is subject to a constant demand for growth and alterations over relatively short time intervals?” If the demand for growth, additions, or renovations is once or twice in the lifetime of the structure, precise and deterministic planning for optimal results is far more reliable. However, responding to constantly changing and unpredictable demands for expansions, or possibly alterations and contractions, probably requires completely different system behaviors.

One limitation of a method using DLA is that the method does not always guarantee to provide an absolute best solution at every time step of growth. This is part of the nature of stochastic simulation. Finding the deterministic optima for multiple steps in advance is a challenge. One way to approach this can be to iterate through numbers of trials to find fitter configurations multiple steps in advance.

Finding scenarios that require gradual growth over time in architecture is a challenge. Except for some urban-scale developments, the scale of physical size and the magnitude of time that it takes to grow have, for buildings, not reached a stage where we require such a design method. In most cases, practitioners can forecast adequate solutions analytically, and are very unlikely to find any kinds of building development that require step-by-step constant improvements in shorter segments of time like Kowloon Walled City. All that said, I would like to anticipate the emergence of architecture that takes advantage of metabolic adaptation as seen in the accretive growth model by continually adapting itself to new conditions as if it were an organic entity.

5.2 Adaptation: Physical Implementation Using Multi-dimensional Optimization

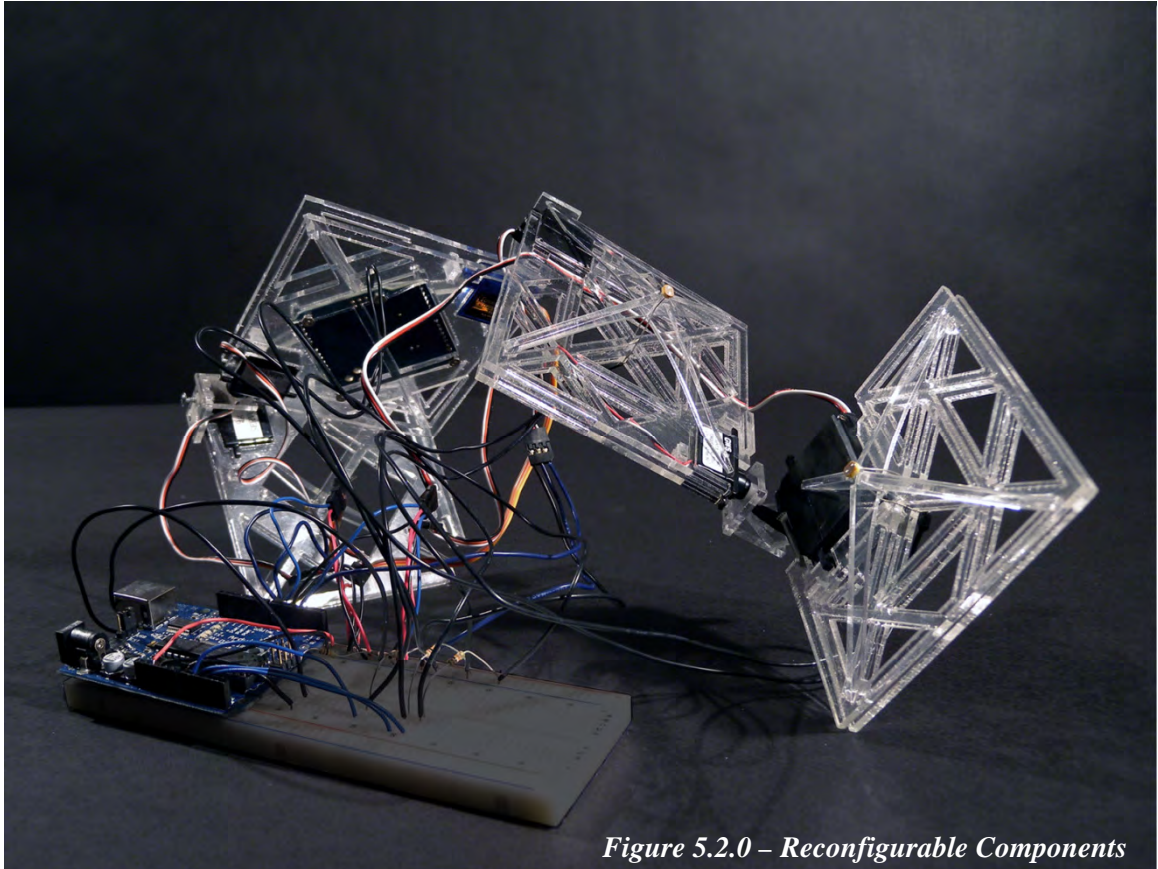


Figure 5.2.0 – Reconfigurable Components

In this section, I would like to propose a scaled prototype of architectural components that can reconfigure themselves into globally functional configurations based on feedback from locally distributed intelligence embedded inside the component. The project aims at demonstrating a design system that can respond to a dynamically changing environment over time without imposing a static blueprint of the structure in a top-down manner from the outset of design processes.

5.2.1 Distributed Systems

Distributed control is one technical strategy to realize a feedback process inside a bottom-up system, and this strategy can be applied to the control of multiple structures. Inputs for this feedback system are fed from separated nodes and can be triggered by participation of independently acting agents with some intelligence. The entire system's behavior is a result of feedback from multiple distributed intelligent sources, and such a system is often called "collective intelligence." The assembly and control of the subunits are governed by the logic of a distributed system simulated by the use of multiple Arduino microcontrollers (Arduino is an open-source electronics prototyping platform; <http://www.arduino.cc>, 2010.) Appropriate geometrical configurations will be computationally derived based on local communications among the components and feedbacks based on physical-environmental criteria such as solar radiation (from various sensors) and on social-programmatic factors.

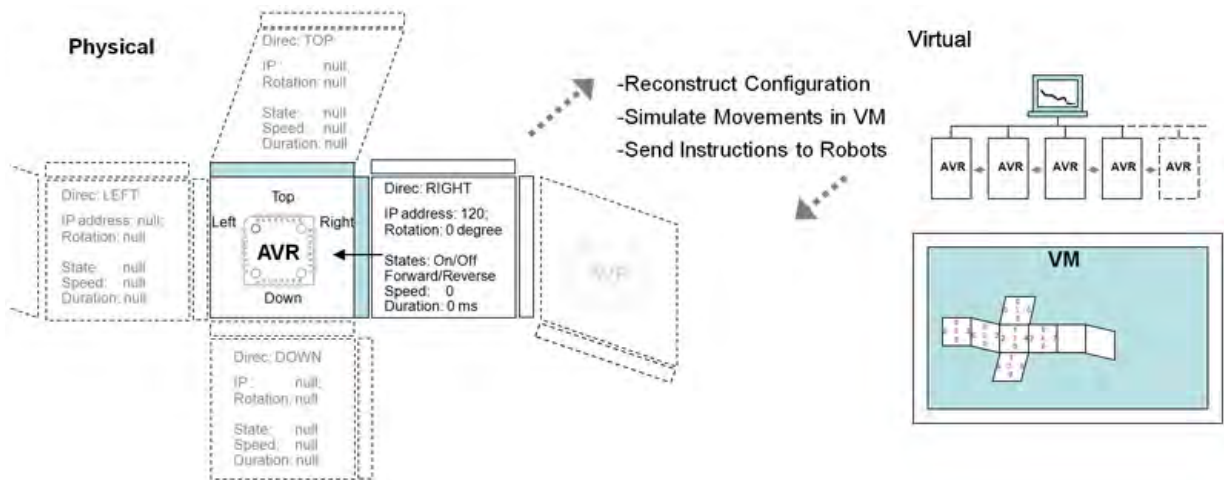


Figure 5.2.1 – Concept of Distributed Controls

The objective of the experiment is to establish a system that can dynamically respond to changes of light source locations in order to increase light exposure by reconfiguring its components' locations and angles toward the light source. Reduction in response time for dynamic reconfiguration is expected from the application of the logic. Each component is connected at the joint with two degrees of freedom provided by a pair of servo motors, which offers sufficient variations in configurations when multiple components are forming clusters. This mechanical set-up allows them to configure all possible patterns in orthogonal geometry as long as all components are contiguous in series. Each microcontroller is responsible for the control of several adjacent components (only two controllers are used for the current experiments), and neighboring controllers can, in principle, send and receive their states, such as their orientations in degree, levels of solar radiation, thermal conditions from various sensors, architectural programs of components, and so on. Based on feedback among the neighbors (informational exchanges among the components), each microcontroller will send a signal to change its components' states in order to locally optimize its condition. Consequently, multiple interactions among the locally defined actions lead us to obtain globally functional configurations rather than a final form being imposed in a top-down manner.

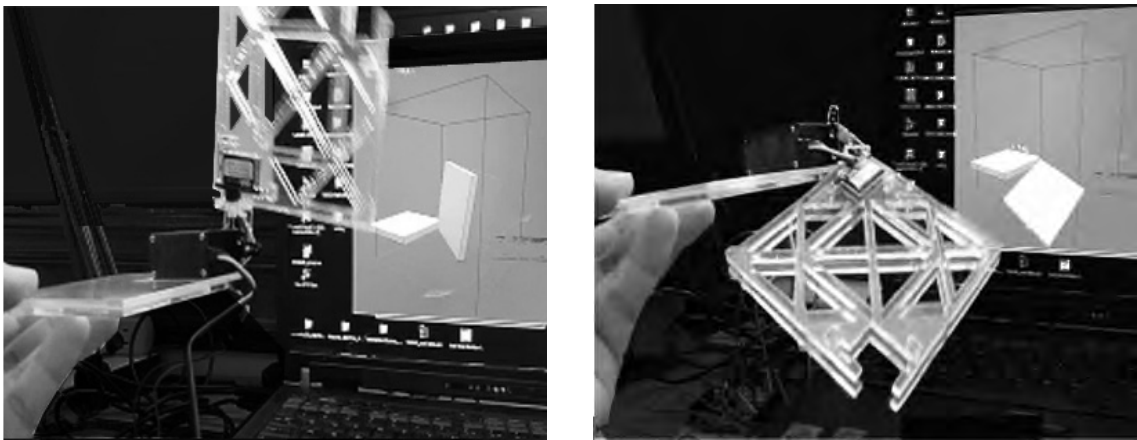


Figure 5.2.2 – Established physical and virtual synchronization using the Virtual Controller Software written in Java/Processing

To construct the present system, firstly, a graphic user interface which can display and control four components' movements was developed using Java/Processing language. This allows bi-directional communication between physical and virtual environments (Figure 5.2.2).

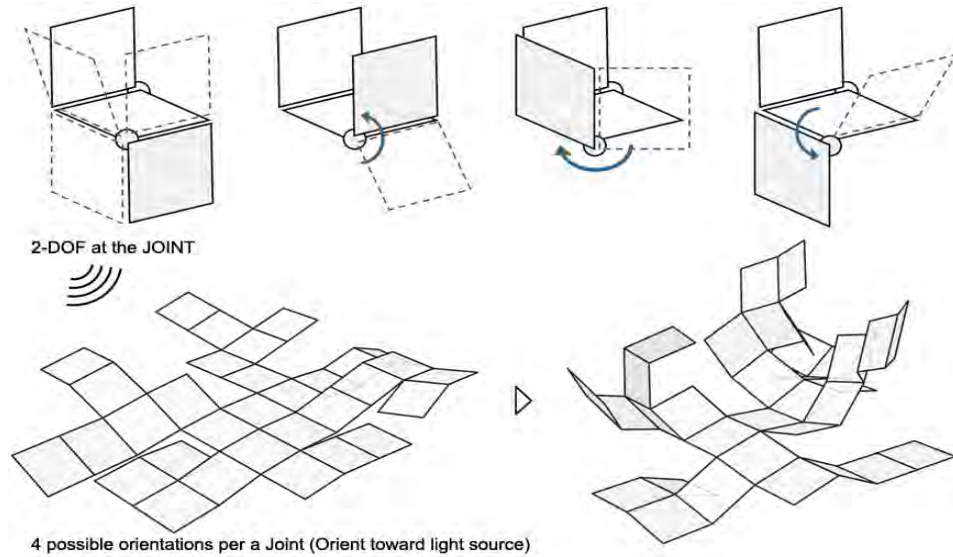


Figure 5.2.3 – Two degrees of freedom at joints. Panels orient toward a light source.

As a starting point of this experiment, each component was supplied with a photodiode (light sensor) to measure the level of solar radiation at the panel surface (Figure 5.2.4). Sensors returned the values to assigned microcontrollers based on the current orientations of the components, which can be varied by rotations of the motors in tandem at the joint. If components change their configurations with different rotation angles, naturally the results from the four sensors will have different values. There are four panels connected in series at three joints, so that there are, in total, six motors to govern all the configuration patterns. In order to find better configurations to maximize average solar exposures for each component's panel surfaces, we have to find ways to derive better combinations of the six rotation angles of the motors. This framework for the problem led to the use of multi-dimensional optimization algorithms.

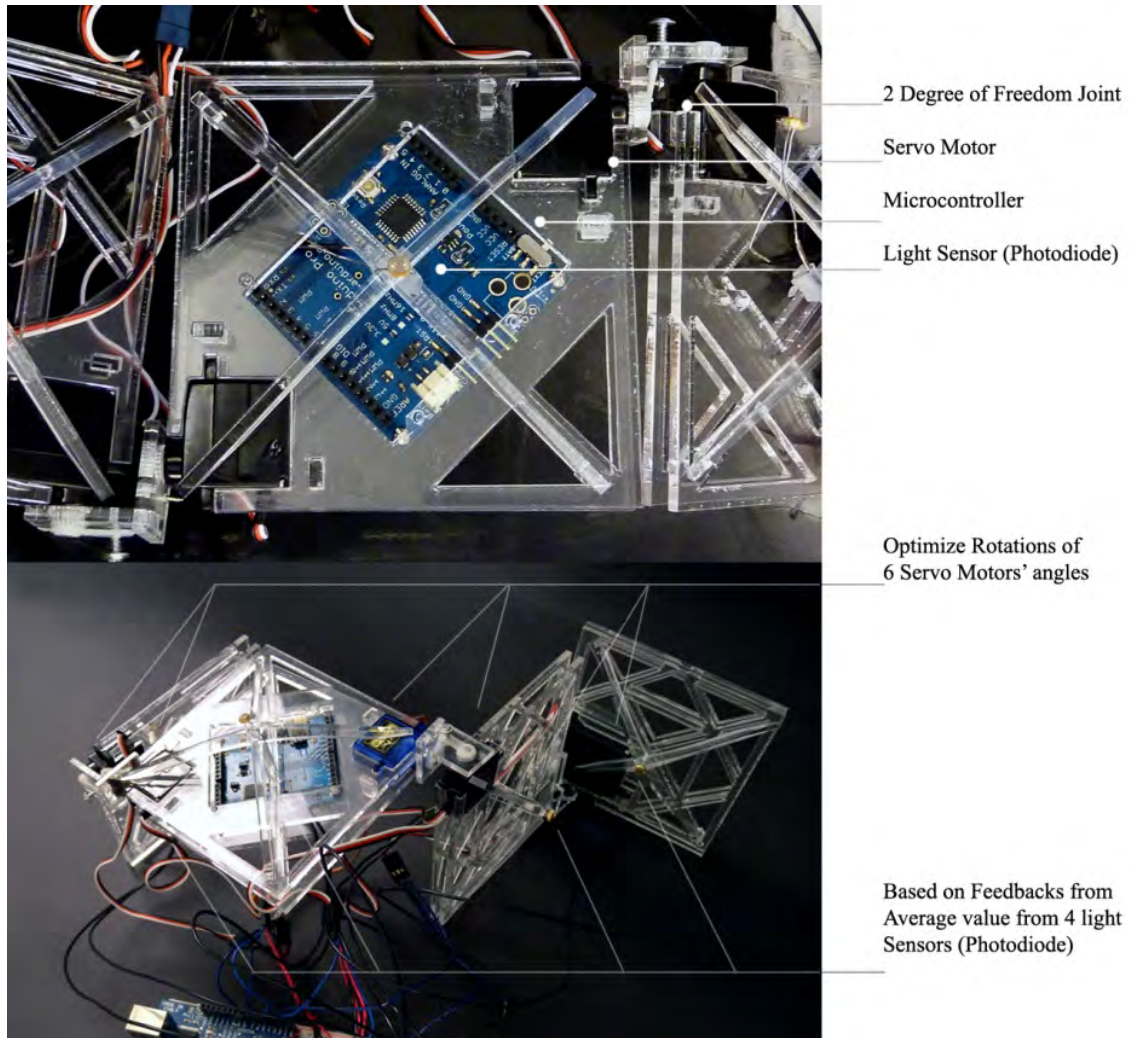


Figure 5.2.4 – Components autonomously find better configurations to maximize light exposure at 4 sensor nodes at the middle of the panel using two different optimization algorithms.

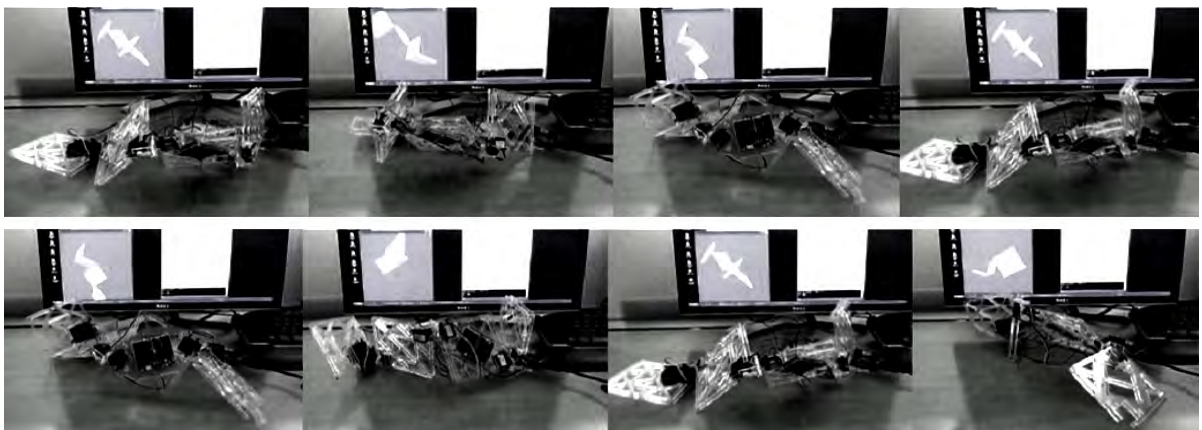


Figure 5.2.5 – Physical and virtual synchronization using the software written in Java/Processing

5.2.2 The Nelder-Mead Method: Physical Implementation

The Nelder-Mead method is a commonly used nonlinear optimization algorithm and is often used for minimizing an objective function in multi-dimensional space. In the case of this experiment, the search is in six-dimensional space formed by independent variables of six angles of motors, and the mechanical components literally become a physical objective function to provide values (average values of four light sensor outputs) which need to be minimized (the lower the sensor value, the higher the light input value). First, the algorithm configures seven different physical configurations to sample light values for each case, which will form a polytope of 7 vertices in 6 dimensions (using the simplex concept). Then, the algorithm will rank them based on sensor values' feedbacks from the physical machine and search for vertices which provide better configurations of new motor angles. The robot will show the best configuration and turn the LED indicator on. The algorithm will repeat the above processes until it stops improving the value above a certain minimum. This method is also nicknamed "the amoeba method" (see Figure 5.2.7) since the way the polytope finds the new vertices and moves towards a better solution inside the multi-dimensional space is similar to the movements of amoebas. In case the directions of light sources are altered, the system will dynamically react to the changes and will run the algorithm based on the values returned from the new condition.

For searches in two-dimensional space, a polytope forms triangles (3 vertices), and Figure 5.2.6 shows examples of amoeba processes applied to a simple 2-D objective function. Each vertex of the triangle (B , G , and W in Figure 5.2.6) has a different value

for the function F (i.e., $F(B) < F(G) < F(W)$). In each iteration of the Nelder-Mead method, the worst point, W , with the largest value from the function will be replaced by a point with a better value by using the Nelder-Mead algorithm. (In this case, the objective is to minimize the value of function F .) Acquisition of the new vertex is based on reflection, expansion, contraction, or multiple-contraction of the current triangle in 2-D space based on the function's returning value of the newly defined vertex. The algorithm finds a better vertex using one of the geometrical transformations of the triangle (Figure 5.2.6).

- Search Process

Search in 6-dimensional space (6 Rotation angles of Motors) \rightarrow find optimal lighting condition from average of 4 sensor (light diodes) values to Optimize minimum $F(\sum \text{sensor}_i / n)$

Initial triangle: $F(B) < F(G) < F(W)$; $R = (M - W) + M$.

If $F(R) < F(G)$

If $F(B) < F(R)$ Then $W \rightarrow R$

Reflection

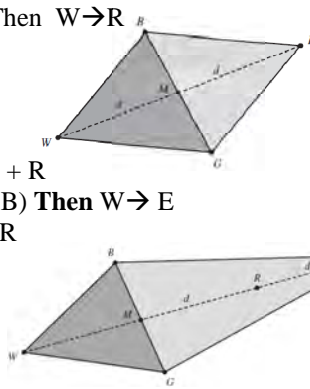
Else

$E = (R - M) + R$

If $F(E) < F(B)$ Then $W \rightarrow E$

Else $W \rightarrow R$

Expansion



Else

If $F(R) < F(W)$ Then $W \rightarrow R$

Contraction

$C = (M + R) / 2$

If $F(C) < F(W)$ Then $W \rightarrow C$

Else

$S = (B + M) / 2$ **$W \rightarrow S$**

Multiple-Contraction

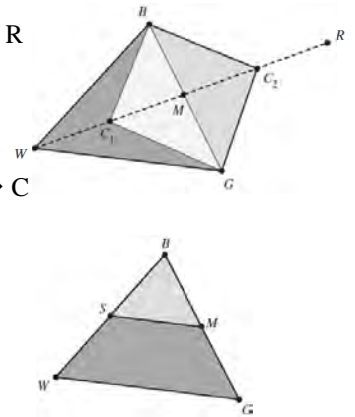


Figure 5.2.6 – Amoeba processes by Nelder-Mead Method Algorithm.

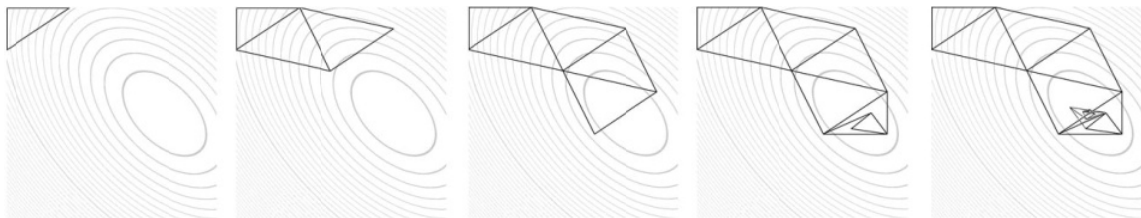
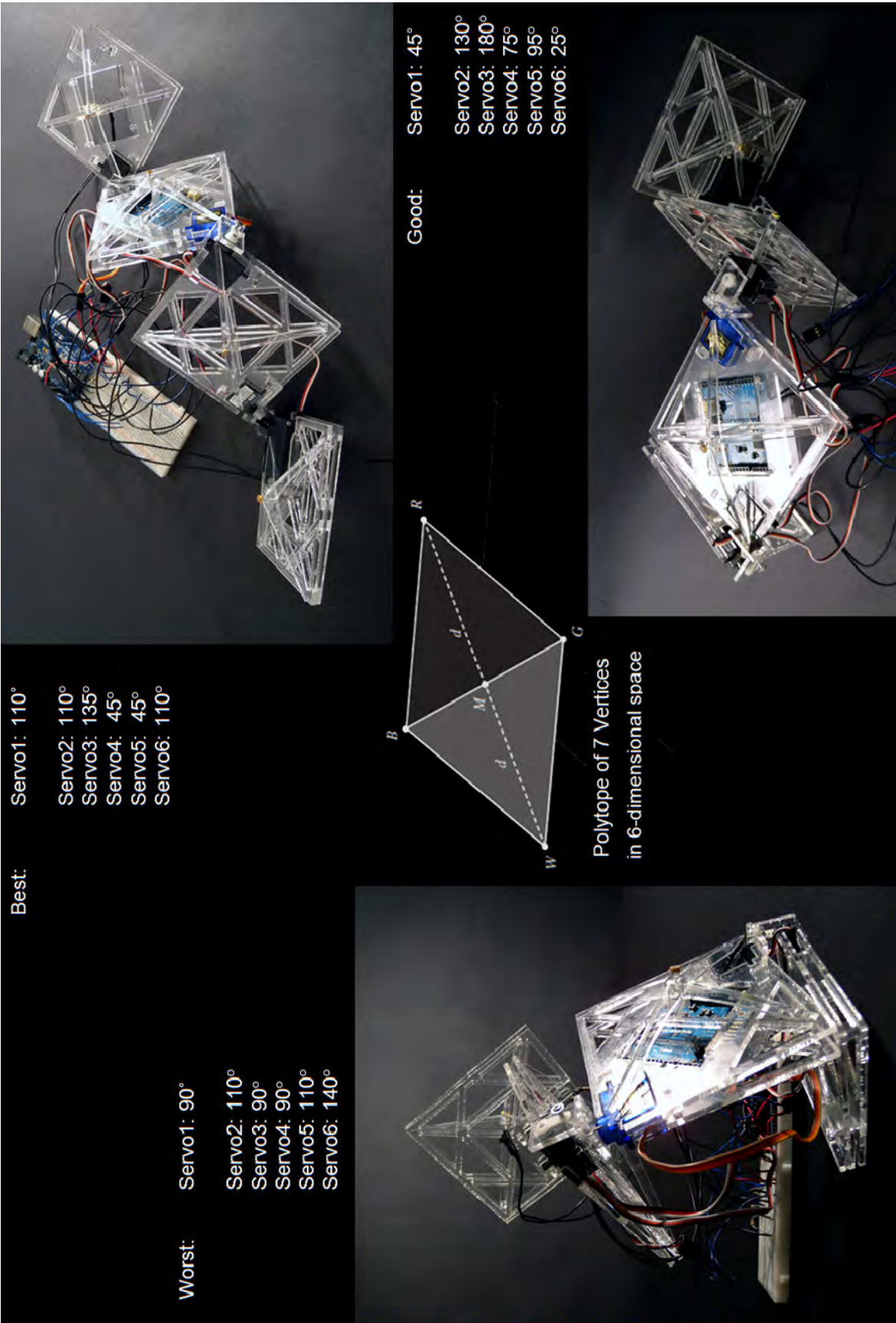


Figure 5.2.7 – Amoeba processes applied to a simple 2-D objective function:

$$f(x, y) = x^2 - 4x + y^2 - y - xy.$$



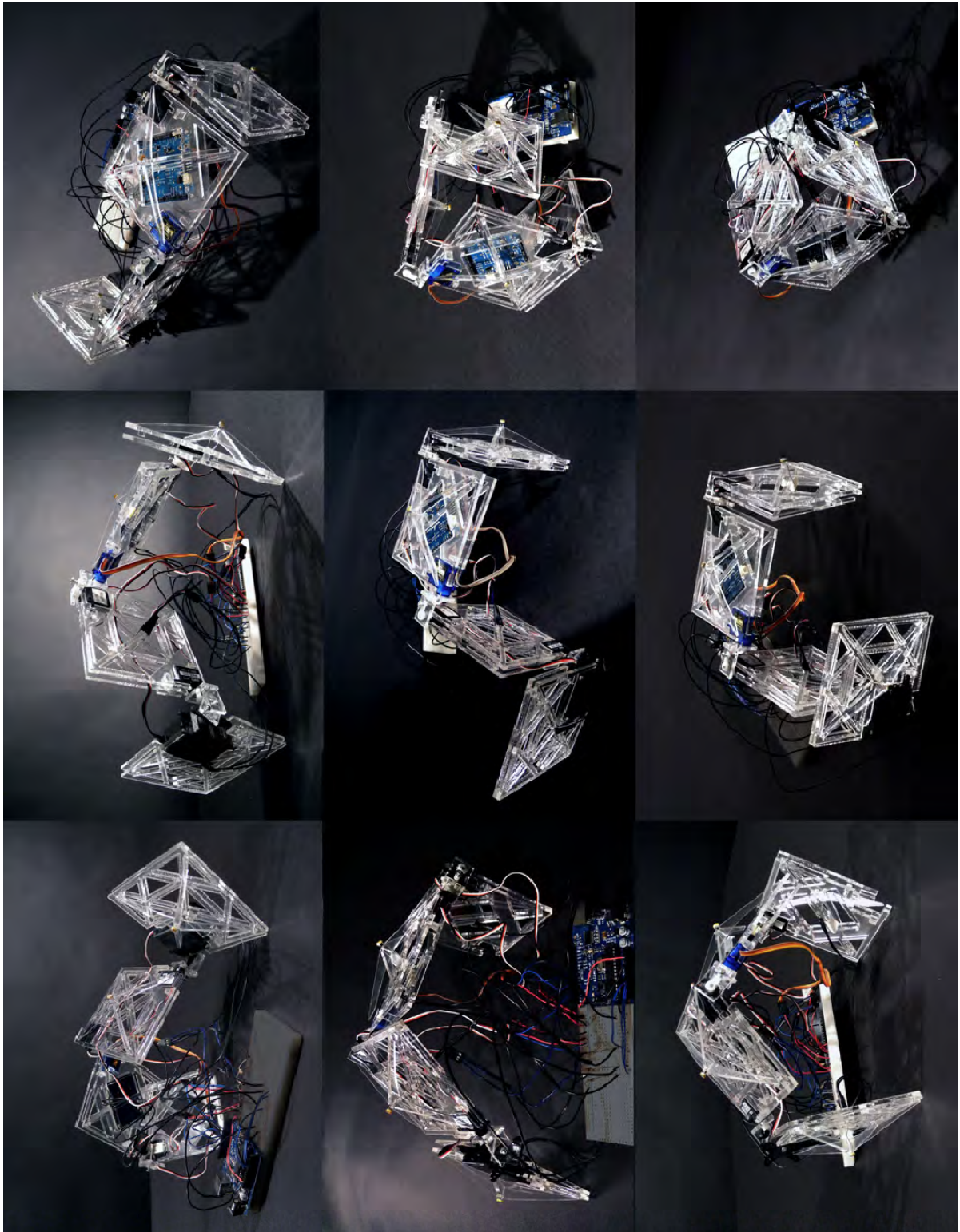


Figure 5.2.8 – Robot tries to find better configurations to receive more light exposure on its four panels using Nelder-Mead multi-dimensional optimization (previous page).

Figure 5.2.9 – The process of dynamic reconfiguration and search (this page).

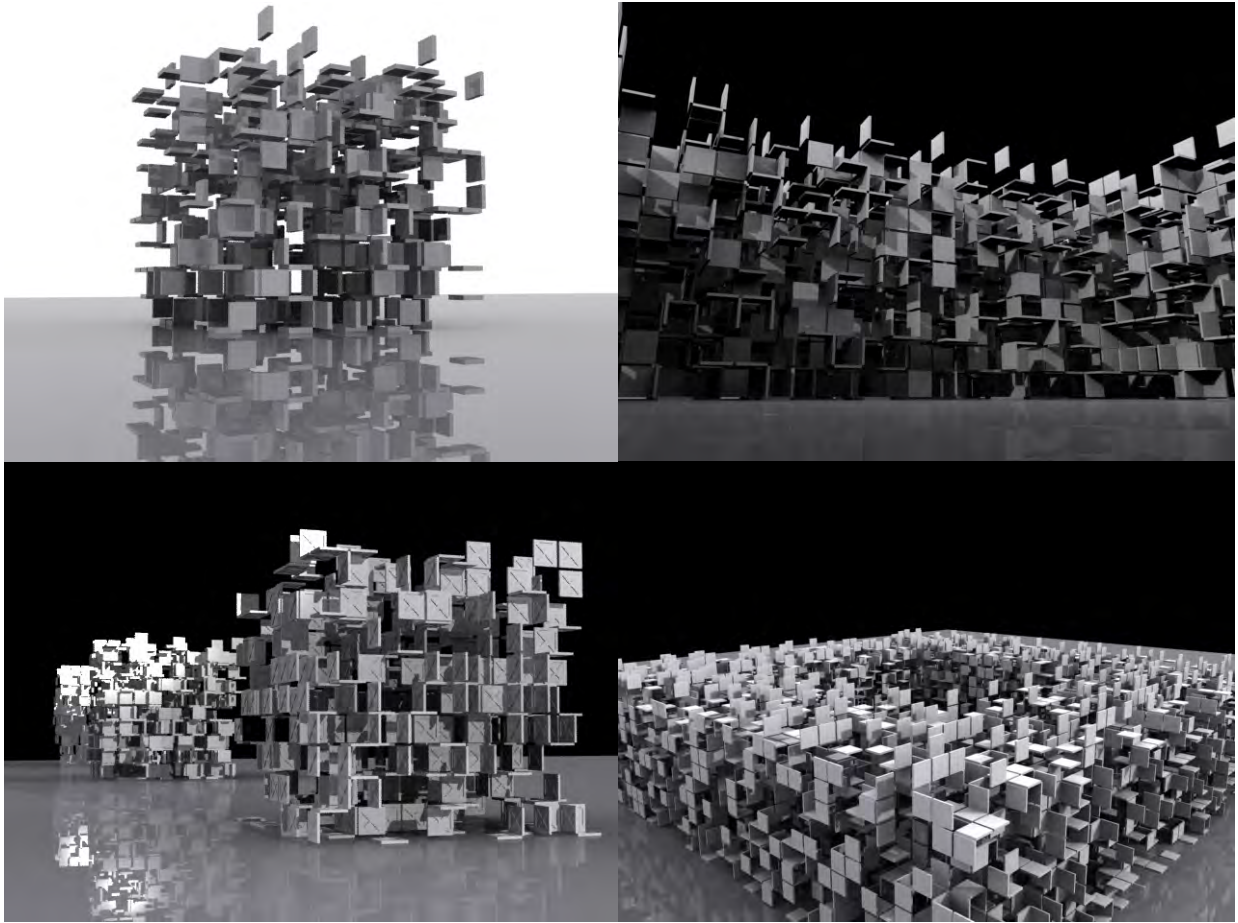


Figure 5.2.10 – Possible future applications using reconfigurable units.

There are many types of multi-dimensional optimization methods similar to the Nelder-Mead method (NM), such as the Levenberg-Marquardt algorithm. The choice of the NM algorithm for the project was merely based on its simplicity of implementation and its geometrically intuitive logic. Other nonlinear optimization methods can be selected for different frameworks of problems in order to gain optimally better performances.

Prior to the Nelder-Mead method, the simpler random search method was tested. In this method, an algorithm rotates each joint continuously in one direction until it stops improving the assigned sensor values for the joint, compared to its former state. Then it

rotates the joint in the other direction to test the improvement. It is a simpler strategy for preventing the system from stagnating at local optima. The results show that the use of the Nelder-Mead method reduces the number of trials needed to find better configurations compared to the simple random search. In addition, using the Nelder-Mead method, responses of reconfiguration to dynamic changes of light source directions are better (which means that fewer trials are required to obtain appropriate orientations of panels for newly defined lighting conditions in changing environments).

Having physical mechanical components be an objective function providing fitness values is a unique and original approach in this experiment. However, it is debatable whether this approach is practically feasible for large-scale architectural applications. Beyond a certain physical scale of application, moving architectural units physically to test different configurations will be inefficient as the weight of the units becomes prohibitive. Furthermore, the numbers of trials that are required to find optimum configurations will increase exponentially as the numbers of components grow. The virtual controller in this project reports the physical orientations of the components. For future explorations to find more practical applications, it would be desirable to have more comprehensive simulation environments that can virtually estimate structural, programmatic, and environmental fitness, including energy calculations, without physically moving the components. Bi-directional control combining the use of both physical and virtual objective functions will allow error corrections between the two environments and appears to be a more promising approach. More specific application examples of this approach could be an architectural shading control that can respond to unknown or un-programmed conditions by using direct feedback from physical

conditions. Another advantage of the application of a distributed system is robustness. Unlike conventional central control systems, failure of several controllers does not imply failure of the whole system, and this type of robustness is promising for applications of architectural elements in exterior or inaccessible areas.

5.2.3 Discussion and Critique

Development of flexible and adaptable architecture has been a recurrent theme among practitioners. There have been several inspirational projects in the past. During the 60s in Japan, Metabolists introduced mega-structures that could constantly grow and adapt by plugging prefabricated pods onto the infrastructural core; however, original visions of metabolic growth and adaptation were rarely realized physically, as the sizes and weights of the pods were practically very difficult to reconfigure. In the 90s, construction automation (Shiokawa et al., 2000) by general construction companies in Japan shed light on the concept of self-reproduction in architecture: architecture that can produce architecture. However, there was still a clear division between assembler and assemblee relationships. Mechanical components that could produce buildings were far from actual livable architectural spaces. Thus they could only repeat, producing an identical or similar building each time, and no future adaptation was available. Finally, some of the speculative researches by computer scientists in recent years have started to show viable prototypes representing self-reconfigurable systems using swarm robotics (Lipson, 2000; Murata, 2006). In architecture, it is our responsibility to consider how these advanced technical concepts can be applied to enhance our living space, and our design processes

may well be on the brink of a necessary transition from conventional methods to methods that require evolutionary processes.

For this project, it is worth noting that the Nelder-Mead method is a heuristic method. A heuristic method is a solving of a problem by iterative processes of trial and error and is intended to find optimal solutions rather than to find a single deterministic solution. Traditionally, we have a tendency to seek and construct an analytical problem-solving framework due to the invisible pressure to find the final and best solution. Finding a single solution, static in time, that satisfies various clients' needs has been a typical architect's responsibility, and generating comprehensive plans as a blueprint is normally anticipated. Conventional design problems in architecture may be more easily reducible to an analytical problem-solving framework, compared to finding optimal solutions over time every step of the way. As can be seen in this project, calculations of dynamic reconfigurations for gradual growth of structure can be fairly extensive. This fact implies that the deterministic analytical means are less adequate where we need concurrent solutions for dynamically changing conditions, and we may need to rely on heuristic search as the complexity of the project increases.

As for the implementation in architecture, it is extremely important to consider not only physical and quantitative issues but also internal and qualitative issues. Environmental issues such as lighting can be quantified and may be resolved to some degree; however, more programmatic issues relating to logistics of architectural planning will need to become a new focal point of research among our profession. An aim of the present thesis is to show how self-reconfigurability can be incorporated into architectural design

processes in order to realize an adaptable growth model through an extremely simplified working conceptual prototype.

The experiments in this section are not at the stage of providing a direct application to existing architecture. In principle, the numbers of components can grow and reconnect to expand the structure to respond to increasing and differentiating spatial demands. Solar radiation was one criterion selected for the reconfiguration; however, various different criteria can be technically implemented in the system. For instance, affinities among various occupancy types and their adjacency relationships can be used as a selection and morphing process of various architectural programs. This selection process can be achieved by cellular automata-based logic, similar to the method introduced in (Arduino, 2010). Allocations of different architectural programs such as residence, office, and retail spaces can also be optimized *virtually* by the use of various simulation programs and *physically* by tracking the movements and behavioral patterns of occupants in the future. Further investigation will be required for implementations of additional architectural applications. The intention of the project has been to clarify the concept of dynamic form-finding technique based on the bottom-up approach through a rather simple and clear form of prototype.

5.2.4 Conclusion

In architecture, few structures have ever been built or conceived based on the active application of distributed systems. Excluding some of the emergent formations of cities on larger scales over longer spans of time, adaptation of distributed systems and

collective intelligence to architectural creations is an uncultivated area worthy of investigation. This project is one such effort to demonstrate a novel design system through a conceptual physical prototype that simulates the concept of dynamic adaptation in architecture.

Chapter 6

Application of Self-organizing Computation: From Prediction to Synthesis

Introduction

Spontaneous settlements in many Third World cities are often regarded as undirected, chaotic, and negative; however, their informal growth patterns exhibit transient and flexible characteristics that can autonomously generate emergent infrastructures. This research investigates the growth logic of informal settlements in relation to natural environmental conditions.

Favelas in Rio de Janeiro are one such example that represents the decentralized dynamics of squatter settlements. Typically, such settlements are located on urban land that has remained unbuilt-on – areas such as deep valleys, river banks, and dangerous slopes – because the squatters cannot afford to live on a safe site. Many of these settlements are triggered by inadequacies in existing infrastructure, and informal

organizational networks can autonomously generate emergent infrastructures on the basis of certain given factors. The settlements' ability to adapt to landscape topology, environmental changes, and radical population growth occurs through self-organizing processes that can dynamically alter goals in accord with necessities and purposes arising from groups of individuals.

This investigation explores the possibility that the “emergent” quality seen in informal settlements is potentially beneficial for compensating for the lack of robustness in our current top-down planning methodologies. Through on-site investigation of chronological settlement patterns, simulation software will be developed from the knowledge acquired from the site using agent-based computation. The purpose of this research is to find a way, by investigating these informal organization processes, to synthesize this novel self-organizing design concept with our existing top-down methodologies.



Figure 6.1 – Favela in Rio today called Rochina. (From <http://irishabroad.blogspot.com>)

The first part of the research will be the development of a simulation tool that can represent the gradual growth patterns observed in spontaneous settlements. Use of a multi-agent system is a valid strategy to simulate the decentralized dynamics of informal settlements. The implementation of agents' behaviors will be based primarily on topographical conditions, intensity of existing traffic, and attraction to the emerging destination locations; the chronological growth patterns of the settlements will be encoded as a motivation for agents to initiate new constructions.

The second part of the research will be applications of the simulation tool to the sites of various informal settlements. The primary focus will be on investigating the correlation between sites' unique landscape topologies and their settlement patterns. Locations and proximity to natural resources such as water, natural greenery, and open land for farming, are key incentives for settlers to initiate new constructions. Maps that represent the chronological development of each site will be produced, and the morphological growth process of the settlements in relation to land form and environmental criteria will be investigated.

The development of a novel system that can simulate the spontaneous growth of cities in precise geometrical detail based on a given landform and environmental conditions will be a contribution to both landscape and architectural computation studies.

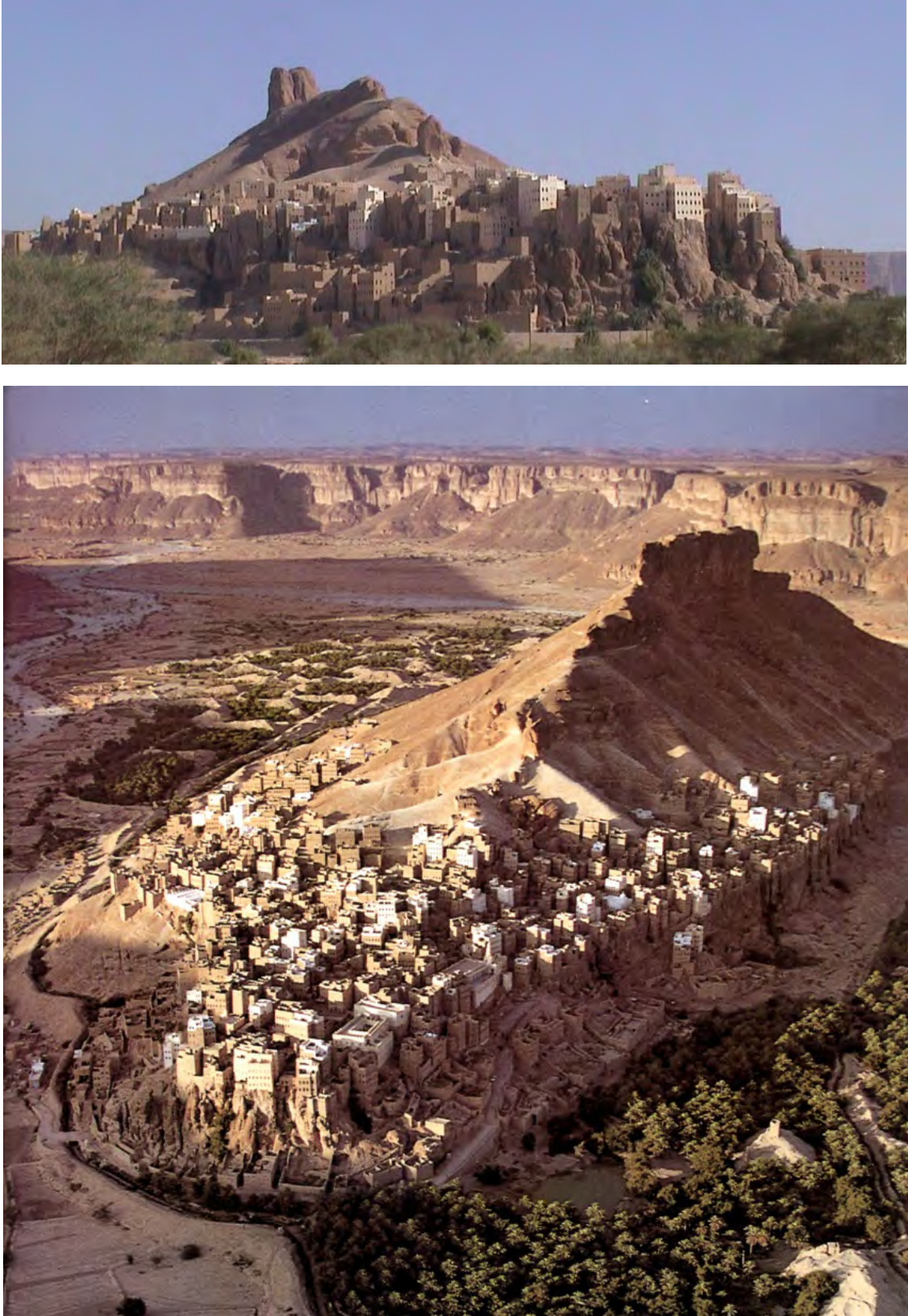


Figure 6.2 – Traditional settlement: Views of a town in Hajjarayn, Yemen (Costa, 1977).

6.1 Objectives

Landscape architect James Corner claimed in “Urban Natures” (2000) that *“new city forms are an amalgam of mobile agents, provisional colonies, and diverse components. They are composed of small units and collectives rather than singularities, and bottom-up organizations rather than top-down orders.”* There is a growing interest in the logic of city formation directed by self-organizing networks and in further utilizing these logics in frameworks for design practice.

There are several existing software applications to generate hypothetical virtual cities. Earlier, I have reviewed works by various researchers using shape grammar, L-systems, and so on. However, most of them only display results at one static time frame and do not portray a gradual growth process in a time series. As discussed above, many design systems require imposing a specific design template based on typological layouts of cities plus primary inputs of data, such as population densities. In addition, the emergent process of city growth (the morphological growth process of cities) is highly influenced by land form and environmental conditions of a site, and not many existing applications are successfully addressing this point.

A crucial difference between the proposed system and previous approaches is that the data of chronological development from the existing site will provide the implementation of agents’ behaviors. The proposed system will not *impose* existing patterns as a design *template*. Instead, the knowledge is acquired from the process of development steps over time, and implementation of the knowledge is conditionally applied in continuous

sequences. Placements of buildings by agents are highly influenced by land forms, climate, proximity to green areas, and water resources. In turn, agents will also be changing the current environment by their placement of buildings, and these changes in the environment will affect the agents' next behavior. By concurrently updating their behavioral patterns according to newly established conditions, the system can form open-loop feedback. This open-loop feedback between agents and environment is necessary for simulating decentralized dynamics of settlements.



Figure 6.3 – Emerging Trail Patterns found at Arthur's Seat, Edinburgh, UK (From GoogleEarth, 2010 and <http://www.panoramio.com/photo/4720638>(top-right))

6.2 Emergent Design System

6.2.1 Development of a Design System: Technical Note

The entire system is written in the Visual C# programming language with the .NET framework (Microsoft, 2010). C# is a modern, general-purpose, object-oriented programming language and provides support for automatic garbage collection. The language is generic enough for any type of software development, and simpler to use compared to its predecessors such as C or C++.

Open Graphic Library (OpenGL) is used as a graphics library for the system (OpenGL, 2010). The OpenGL is the industry's gold standard for a three-dimensional graphics library and is the most widely used today. In addition, the Open Toolkit (OpenTK) is used as a low-level C# library that is a wrapper for OpenGL (OpenTK, 2008).

6.2.2 Environment

The system's fundamental components are terrain, agents, and buildings. The system is intended to simulate the gradual development of human settlements based on topographical information as a primary input. The system assumes that the development starts from unoccupied empty terrain, and wandering settlers' behaviors are simulated by the computational agents. The buildings and streets (trails) will be gradually inserted by the agents as a part of their behaviors.

1) Terrain Generation

Landform is a key original input for the system. Geometrical conditions of landscape are described as a grid of triangulated patches. I set up a uniform grid of point locations in an xy-plane based on user-selected grid size as an interval between adjacent points. Only the heights for each point location vary, according to the terrain that is under consideration. This setup can produce homogeneously gridded cells on a terrain surface, and later it becomes easier to record activities at local conditions of a surface using equally divided cells. These cells, or patches, can store local information about traffic intensity of agents (settlers) and local acuteness of a terrain (slope) at the cells. The terrain under consideration can be produced by the system using the random midpoint displacement method as an artificially created test sample surface, or it can be imported from any geometry from an external environment by using the following features.

2) Importing Surfaces

The system can import any surface geometries created inside of any CAD software environment as long as they are saved in the DirectX (.x) file format. DirectX is one of the standard file formats for meshed surface geometries developed by Microsoft Corporation. The X-file contains information for all vertices, surface normal vectors at all vertices, faces, and textures on the faces in a Cartesian coordinate system. My system can read this information and recreate the geometry using a user-defined model scale factor and a grid size. Based on these two parameters, representation of an original surface can be approximated by sets of triangulated patches in my system. In my system, choosing a smaller grid size relative to a model scale will produce a higher resolution surface. The vertices inside the DirectX file are not always arrayed in a uniform grid in the xy-plane,

so the system alters the locations of vertices based on the user-defined grid size while maintaining the original geometry of the surface. This feature allows users to import any topographic conditions created or found from any other software environments, such as Google Earth. Google Earth has become a good source for topographic information on existing sites, and its geographical information can be exported to X-file format through several different CAD software applications, including SketchUp and Rhinoceros. The system can also save states of the environment separately. Information for the states includes agents' locations, generated buildings and streets, the current state of traffic intensity at the site, seed numbers used for the random number generators, and so on.

3) Random Midpoint Displacement Method

Besides the above importing feature, I have prepared a terrain generator that can provide an artificially created terrain using fractal geometry. This functionality promptly produces a sample test surface that can give an impression of natural terrain, and a global geometry of the output surface that is to some extent controllable by defining the locations of a few initial seed points. I used random midpoint displacement method to generate fractal geometry. At first, this method produces midpoints using the initial seed points. For example, seed points can be four corner points or a grid of points at uniform intervals in an xy-plane with various heights. I set the random number range and generate random numbers in that range. Then the method displaces the heights of those midpoints by those amounts. The method repeats this process recursively until it satisfies a resolution of the specified grid size. One key factor here is that the range of random numbers needs to be updated based on a certain rate per generation as we apply this logic

recursively. I reduced this range in proportion to the generation to maintain the natural impressions provided by fractal images.

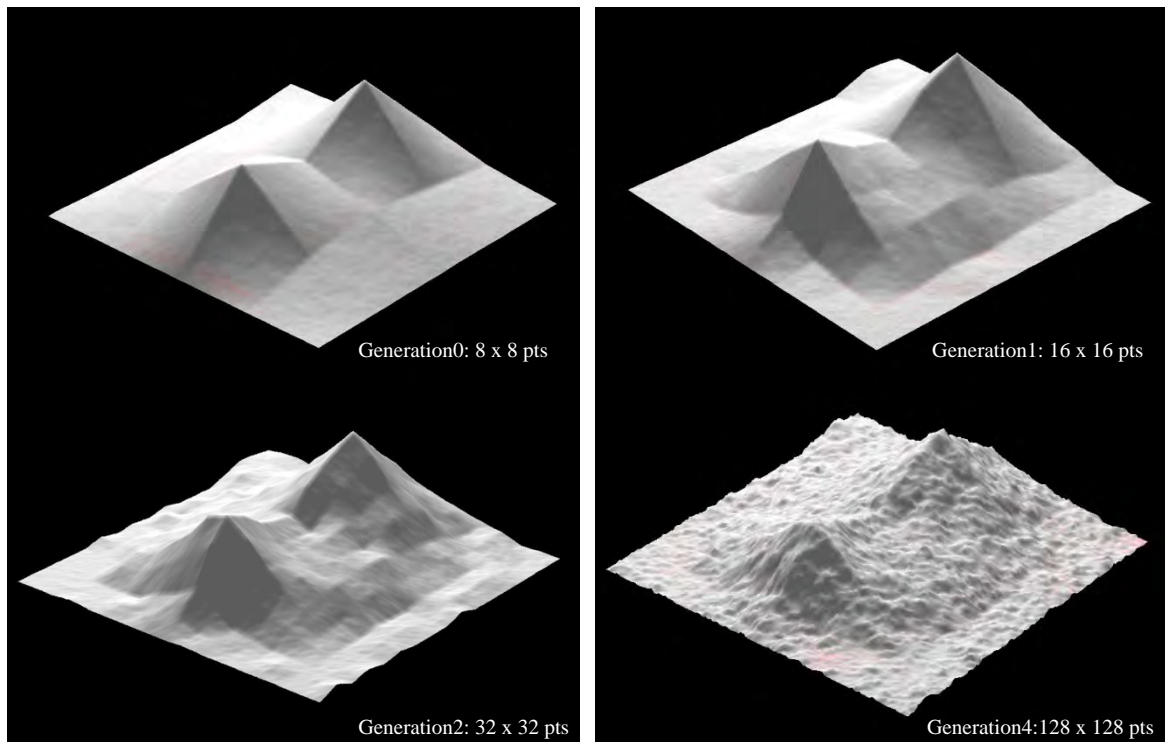


Figure 6.4 – Fractal Noise: Terrain with various degrees of noise.

4) Noise Functions

Some geographical information from external environments does not provide sufficient terrain detail. For instance, surfaces from Google Earth may not provide enough detail for the scale and resolution we want. Some imported surface geometries lack such details and often look overly smooth or overly rough. I have designed two noise functions that can add some realistic irregularities to those imported geometries. These functions apply irregularities in local scale while maintaining a global geometrical condition of an original input. Regardless of the global geometry, agents' (wandering settlers') behaviors are influenced by local conditions of a terrain, and slight perturbations in landform may cause different results for street and building generation by agents.

The first noise function simply uses random displacements for heights of vertices forming a triangulated surface. But the range of this random displacement is carefully set within the range of a given original surface height, so that the result can still maintain the overall characteristics of the original surface.

The second noise function uses the aforementioned midpoint displacement method on a given surface with a user-specified interval distance for the sampling of the original surface vertices. This method uses interpolation of points between the points that are specified by users. The use of the fractal geometry is advantageous due to its natural impression; however, this method skips some of the original source points when it interpolates new points. Hence, it is a trade-off between retaining accuracy to original information and producing a realistic appearance. Some original source terrains are deficient in detail, and fractal noise is a good solution for these cases.

6.2.3 Agents as Wandering Settlers

The original environment of the system is completely vacant: unoccupied. The first settlers arrive at a site from some distance away, at some point in its history. Some settlers might be merely passing by the site, but some may settle down somewhere inside the site. As more settlers migrate through the site, trails gradually emerge. Along these trails, people find preferred locations for their shelters and begin settling down to form clusters of dwellings. Some trails start to bundle together to form arteries, and some branch out to form a complex network of passages. At this stage, settlers are no longer randomly wandering migrants seeking temporary shelter. Instead, they act as

heterogeneous self-driven agents based on clear objectives of their own and travel through these newly emerging cities. Spatiotemporal developments of the site are organized by the settlers' behaviors; however, the environmental changes induced by the settlers also simultaneously influence their behaviors. This co-evolutionary development between the settlers and the environment is one of the unique characteristics of the system. In this section, characteristics of agents are explained in the following subsections: time scale, attraction to slope, slope definition by agents, traffic intensity and chemical trails, chemical-reduction rate (decay rate), diffusion rate, attraction to destinations, and direct path systems.

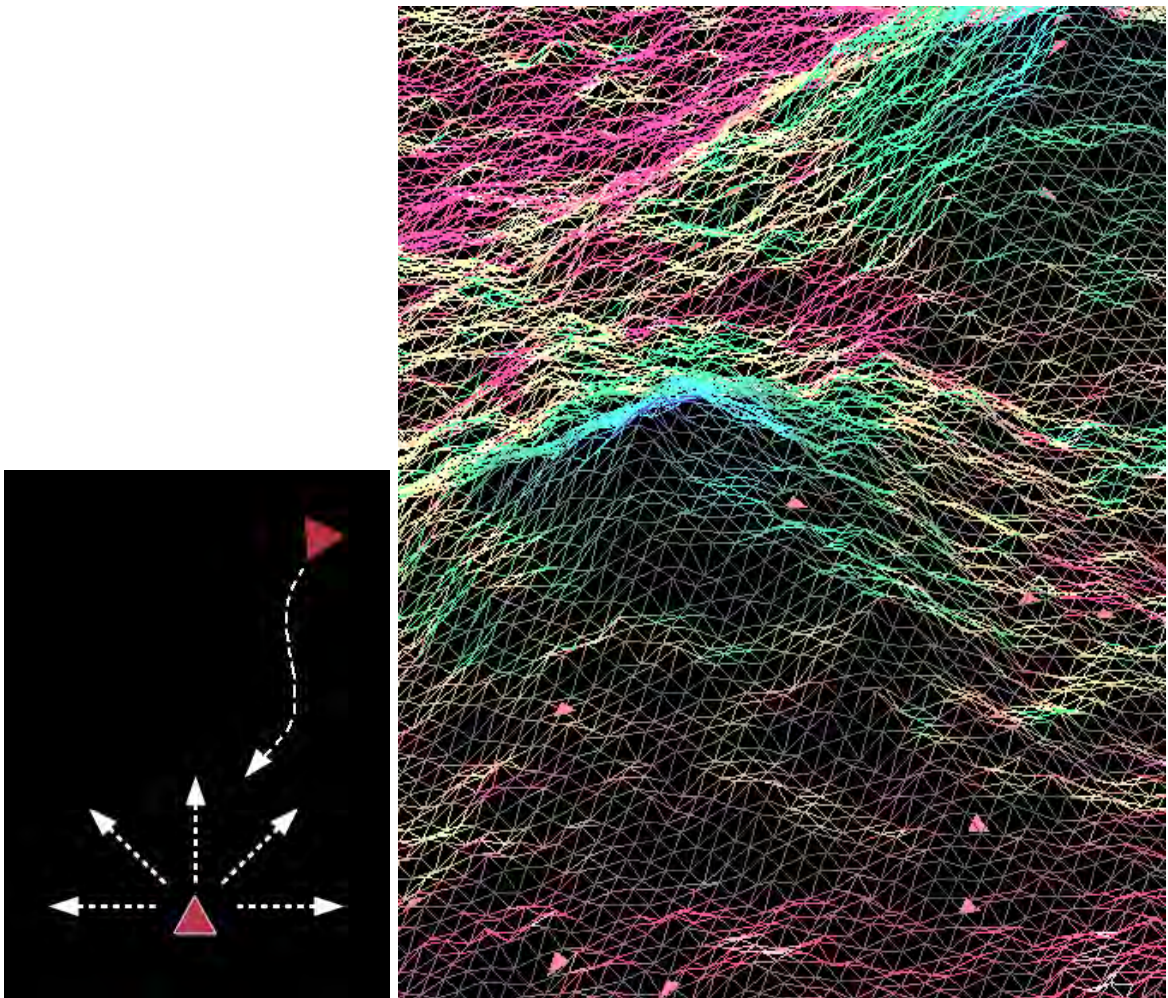


Figure 6.5 – Agents traveling on a wire-framed terrain. Colors indicate frequencies of traffic.

1) Time Scale:

The system has its own scale of time, and it advances its world using discrete time stepping. This is a typical strategy for time-based computer simulation. As seen in Chapter 4, finite difference methods or Euler methods use the same concept of discrete time stepping. Cohesion between the unit time frame of the system and time in the real world is not strictly defined at this stage in the software's development. Scales of time and space have become a critical issue for the project, and one that seemingly can be understood only empirically by comparing the results with actual precedents. I revisit this issue at the end of this chapter.

Settlers are implemented as computational agents in the system. Agents can freely move inside the given environment, yet their movements are restrained on the surface of the terrain within the boundary of the surface. Each agent possesses its own location in a Cartesian coordinate system, and a function to derive a height surface value on the z-axis relative to an agent's x and y coordinates is provided. Agents also possess their ages and their steering directions as heading vectors. Their walking speeds are set based on the terrain surface's unit grid size. In other words, this grid size is the factor to determine the scale of terrain relative to the size of an agent/walker. Later in the chapter, various factors affect this heading vector and influence development of the world inside the system through the agents' behaviors.

2) Attraction to Slope:

Agents are initially equipped with a cone of vision and some level of perception of local terrain conditions. This includes the ability to perceive local steepness of the terrain and intensity of the traffic at the current agent's position. These characteristics will be explained in detail in a later section. Since there is nothing but the landscape at the outset of the process, their movements are simply influenced by the local conditions of the site.

3) Slope Definition by Agents:

When agents calculate the slope of a surface in the direction they are heading, how local this measurement needs to be is a critical issue for the resulting trajectories of agents. Even though some surfaces may have a relatively flat profile overall, they might have bumpy textures locally. Or, vice versa, a smooth local surface might be part of an overall hilly landscape. I defined a parameter called *slope-distance* for the agent. "Slope-distance" defines the point of measurement from an agent's current location. A larger slope-distance value allows agents to apprehend the overall profile of a slope, and a smaller value gives a more myopic view of a slope. There are several ways that agents perceive slope conditions. One is simply using a fixed slope-distance for all measurements. Another approach is to measure slopes at a certain interval up to a selected maximum distance from an agent's current location, and the result is gained by averaging these measurements. This approach is one way to avoid the measurement of a slope being too specific to a certain fixed scale. A third approach is to measure slopes between points at a certain interval along a heading direction of an agent, and the result is gained by averaging these measurements. This approach measures the smoothness of not only the

immediate slope but also the sequence of a potential trip in the heading direction of the agent. However, this does not measure slopes relative to various scales.

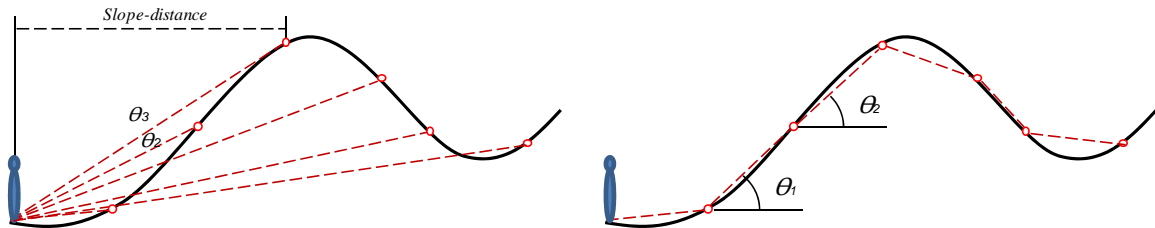


Figure 6.6 – Different ways to perceive slopes by agents. ($Ave. Angle = \frac{\theta_1 + \theta_2 + \dots + \theta_n}{n}$)

The selection of a heading direction is as follows. An agent has the ability to detect local slope of the terrain in all directions around itself, and agents have a “maximum tolerable slope” angle that they can walk on. If the angle of slope is steeper than the maximum value, agents will avoid this direction. Agents compare slopes in all directions in the specified range of the heading direction angle and select the direction with the least steep slope: gentle slope. If all checked directions have relatively similar slope angles, then the agent chooses a randomly selected direction. Once the direction is set, it steps forward and checks the slope angles again at its current new location. Iterating the preceding procedure forms an agent’s ambient movement pattern.

```
void Agent-Step-slope( ){
    //Check slope within -45 to 45 heading range using
    //slope-distance. Choose the least steep direction
    If (all directions have relatively same slope)
        Randomly select a direction from a range;
    If (least slope > max-slope)
        Turn backward direction in random range;
    Proceeds one step forward in selected direction;
}
```

The canonical agent’s gentle slope seeking algorithm

4) Traffic Intensity: Chemical Trail

The terrain is subdivided by the aforementioned grid size, and in this system, the frequency of agents passing through each subdivision of a surface are recorded and stored inside a two-dimensional array variable called “chemical intensity.” Use of the term “chemical” originates by analogy with ant foraging behavior using pheromones, and this part of the algorithm of the system was inspired by the ant colony optimization (ACO) technique (Dorigo, 1992). As more agents pass over any given surface area, this location’s chemical intensity becomes higher. Agents release chemicals with a certain rate called chemical-rate and the rate is set to 0.001 at the beginning. The system can display these values from each subdivision surface visually by graduation of color tone from red (denoting high) to blue (denoting low). Hence, the red area on the terrain indicates an intensive traffic area (high frequency of path use) and becomes an area of potential population concentration. This method can represent a network of passage ways with raster-based graphics (instead of vector-based graphics). Raster-based methods are well suited for representing gradual emergence or extinction of passages in a time series. By contrast, vector-based representation allows one to show only two states, existence or non-existence; raster-based representation can represent degrees of intensity.

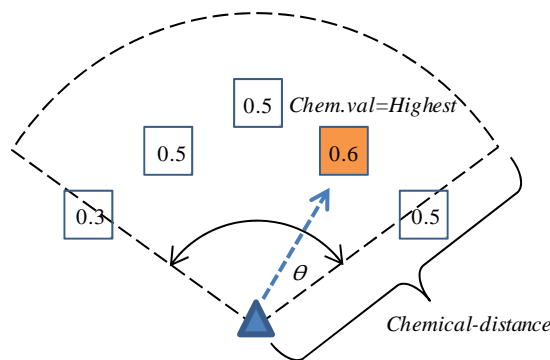


Figure 6.7 – Agent’s cone of vision and chemical value checking mechanism.

As agents gradually find more comfortable routes to walk on based on the above evaluation of slopes, some areas of surface are more likely to be used by the agents as a trail. Using the traffic intensity of the agents, one can visually recognize the emergence of paths. Once these trails have become visible, there will be a shift in agents' behaviors. They no longer need to find their immediate next moves in momentary fashion. Instead, they can recognize some of the paths that are more used by others and become attracted to these popular routes. This behavior accelerates the use of specific circulation paths, and eventually triggers the production of streets and arteries.

```
void Agent-Step-chemical( ){
    //Check chemical level around an agent
    float valbest =0;
    float dircbest=0;
    //Look at -60 to 60 degree range using Max-chemical-dist
    foreach(-60 < dirc < 60){
        for(int i = 1; i <= MaxChemdist; i++){
            value += CheckChemical(dirc, i)*(1/MaxChemdist);
        }
        If(value > valbest){
            Valbest = value;
            dircbest= dirc; }
    }
    //Choose the direction heading the most frequently
    used patches;
    Proceeds one step-forward(dircbest);
}
```

The canonical agent's chemical-seeking algorithm

Similar to the way agents perceive the local steepness of the terrain, agents have a cone of vision to evaluate chemical intensities in their neighborhood areas. Maximum-chemical-distance is the parameter used to define the size of the cone of vision to measure the intensities. In each direction within the range set by the angle of the cone of vision, an agent checks values of intensities at every subdivision surface by incrementing the distance from its position until it reaches the maximum-chemical-distance. These values of intensities are accumulated and compared per direction. The direction that

scores the highest intensity value is selected as an agent direction vector, as the direction has been the most heavily used among all other directions.

The difference between chemical intensity values and slope values is that the measurement of chemical values is cumulatively executed, whereas slope measurement can be executed at the discrete fixed distance from each agent. (The slope measure can be executed cumulatively by setting a maximum distance for the measurements.)

5) Chemical-reduction Rate (Decay Rate)

This chemical value is a dynamic variable of a terrain surface and changes during the course of development. There is a chemical-reduction-rate (decay rate) for this value. Sites of intense traffic or population concentration can shift and move during this development. Trails that were once heavily used but no longer are may disappear. If trails are left unused for long periods of time, their attractive forces will gradually diminish. The system advances every finite step of time, and for every step of simulation in time, the reduction rate is applied to the intensity value of all subdivision surface areas. The chemical-reduction-rate of 1.0 is a constant condition where no trail disappears once it has been created. A chemical-reduction-rate of 0.9 means that the traffic intensity of every position will be reduced at the rate of 0.9 per discrete time step. For example, later in this chapter, I will introduce cases where trails gradually change their topologies in order to avoid steep slopes, thereby producing detours. (The chemical-reduction-rate is analogous to an evaporation rate of pheromones in ant foraging behavior, and in computational interpretation of this behavior, ACO uses this value as well [Dorigo, 1992].

6) Diffusion Rate

The trails of agents are represented as nested values of subdivided terrain surfaces (patches). In reality, these agents represent human pedestrians, and traces of trails can be suspected by agents at locations near these trails. These traces are perceptible by agents at adjacent patches. In order to implement this characteristic, the chemical intensity of patches diffuses into immediately neighboring patches at a specific rate of diffusion. If the diffusion rate is 0.03, the original chemical value of a patch is multiplied by 0.03 and added to the patch's eight surrounding patches in a single time step. This operation transmits original trails' chemical effects to nearby regions, and attracts others in the region.

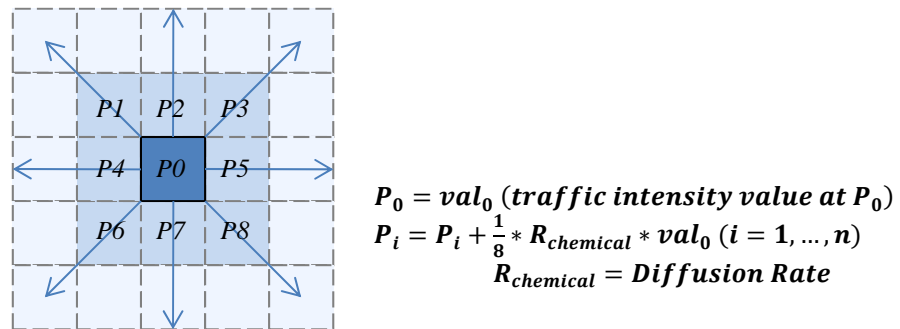


Figure 6.8 – Process of Chemical Diffusion and a diagram.

7) Attraction to Destinations:

The preceding concepts, slope and traffic intensity, are locally defined evaluation factors for agents' heading directions. Nevertheless, the factor that attracts agents' headings can be globally defined as well. Providing stationary points of destinations for an agent's trip is one way to accomplish this. These destination locations can be defined from the beginning, or can emerge later based on the gradual growth of the system itself. A destination can be a point of population concentration such as a city center. Destination

locations are remotely located coordinate points that are stored inside each individual as predefined knowledge. They can be recognized without using any of the previously mentioned sensing features. An agent's heading direction vector can be simply derived by defining a unit vector from an agent's current position to a destination location. This straight line between an agent and a destination forms the shortest path between two points. When agents tour around several destination points one by one, and if they are directed only by attraction to these destinations, agents' trajectories will form a direct path system.

```
void Agent-Step-destination( ){
    //get vector from agent position to dest-city
    If (no destination)
        Vector3 Vdest = (0, 0, 0);
    If (destination exists){
        Vector3 Vdest =
            (Cities[Agent.dest].position - Agent.position);
        Vdest = Vdest.normalize();
    }//Proceeds one step forward in selected direction;
    Agent.position += stride*Vdest;
}
```

The canonical agent's destination-seeking algorithm

8) Direct Path Systems:

A direct path system is produced if each point is linked to another via the shortest distance or route (Schaur, 1991). In this system, a direct path emerges as a result of global configuration if agents are only directed by their attraction to destination points. Here, I would like to note that it is not necessary for the system to impose any knowledge external to the system to achieve a direct path.

So far, I have reviewed three primary factors that govern agents' heading direction vectors. The three primary factors are attraction to gentle slope, attraction to traffic

intensity, and attraction to destinations. From this point on, these three factors will be focused on as primary motivations for agents' physical movements. They can be independently applied to each agent, but they can also be applied together simultaneously to each agent by assigning different weights for each resulting directional vector for the three different attractors. These three weight factors are named slope-factor (**s-factor**), traffic-intensity-factor (**t-factor**), and destination-factor (**d-factor**).

$$\mathbf{V}_{\text{agent}} = \mathbf{s} * \mathbf{V}_{\text{slope}} + \mathbf{t} * \mathbf{V}_{\text{traffic}} + \mathbf{d} * \mathbf{V}_{\text{dest}} \quad (s + t + d = 1.0; \ 0 \leq s, t, d \leq 1.0)$$

A normalized sum of three vectors weighted by the above-mentioned factors produces the heading vector of an agent. By manipulating the proportions of these three factors, agents' behaviors can be directed. For example, weight factors of $s=0$, $t=0$, and $d=1.0$ would produce a direct path system among selected destination points for agents. These values are also dynamically changeable parameters based on changing environmental potentials. In the following section, a simple model with d-factor and t-factor will be looked at, and later I would like to introduce s-factor into this base model.

```
void Agent-Step() {
    Vector3 Vdest = Agent-destination-vector();
    Vector3 Vtraffic = Agent-chemical-vector();
    Vector3 Vslope = Agent-slope-vector();

    // FORMULA //
    Vagent = S_factor * Vslope + T_factor * Vtraffic + D_factor * Vdest ;
    Vagent = 0.05(stride) * Vagent.normalize();
    Agent.position += Vagent ;
}
```

The canonical agent's step-forward algorithm

6.2.4 T + D Model

In this section, I select three predefined stationary points as destination points for agents in a completely flat environment (3 cities). This configuration creates a triangle on an open, level plane, and agents will tour around these three points at the vertices of an isosceles triangle. Agents possess a single destination point at a time, and once they reach that destination, a new destination will be randomly assigned from among the remaining destination points. This is a strategy to make virtual itineraries for agents. With this simple setting, various different values for parameters are used to study different configuration results: chemical-reduction rate (decay rate), diffusion rate, maximum-chemical-distance, t-factor, and d-factor.

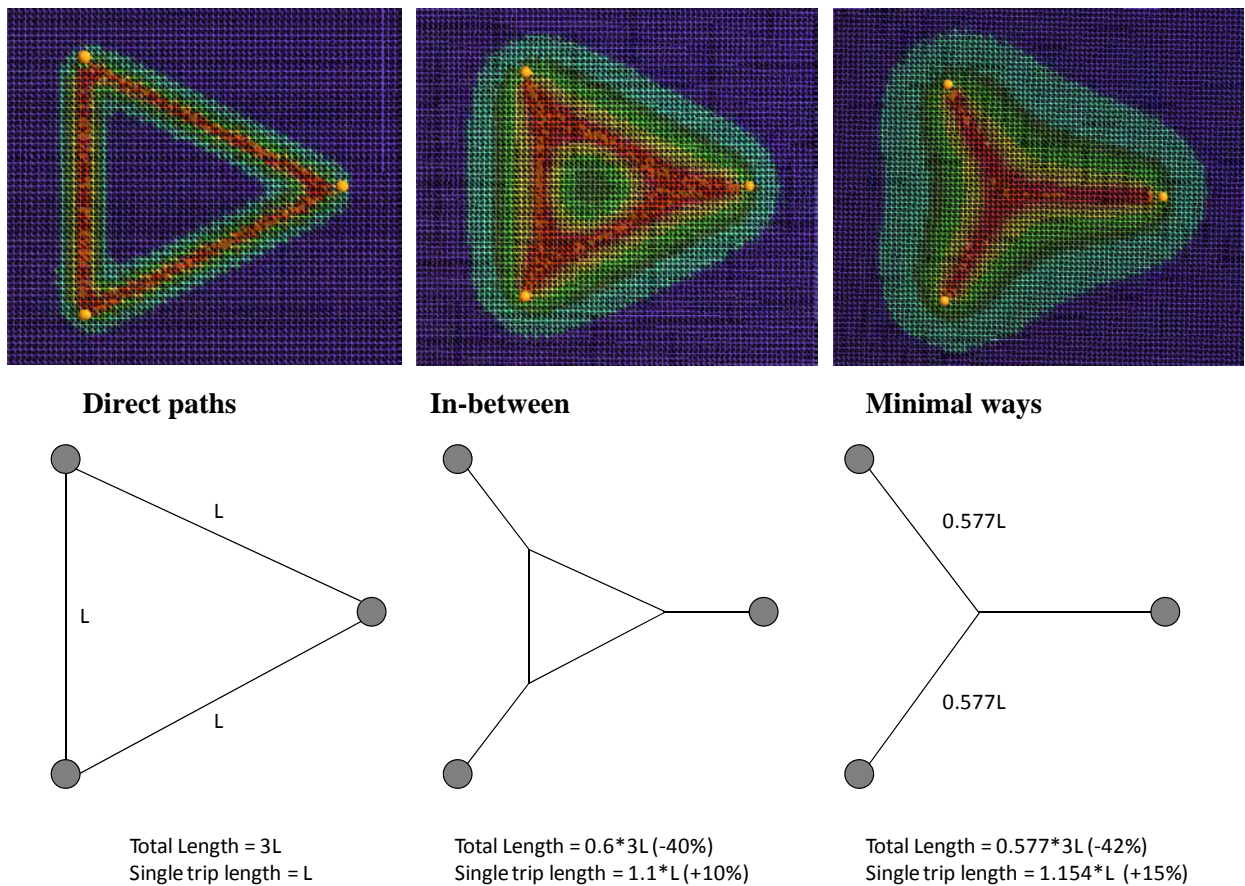


Figure 6.9a – Simulation on 3 Cities: Transition from Direct paths to Minimal ways

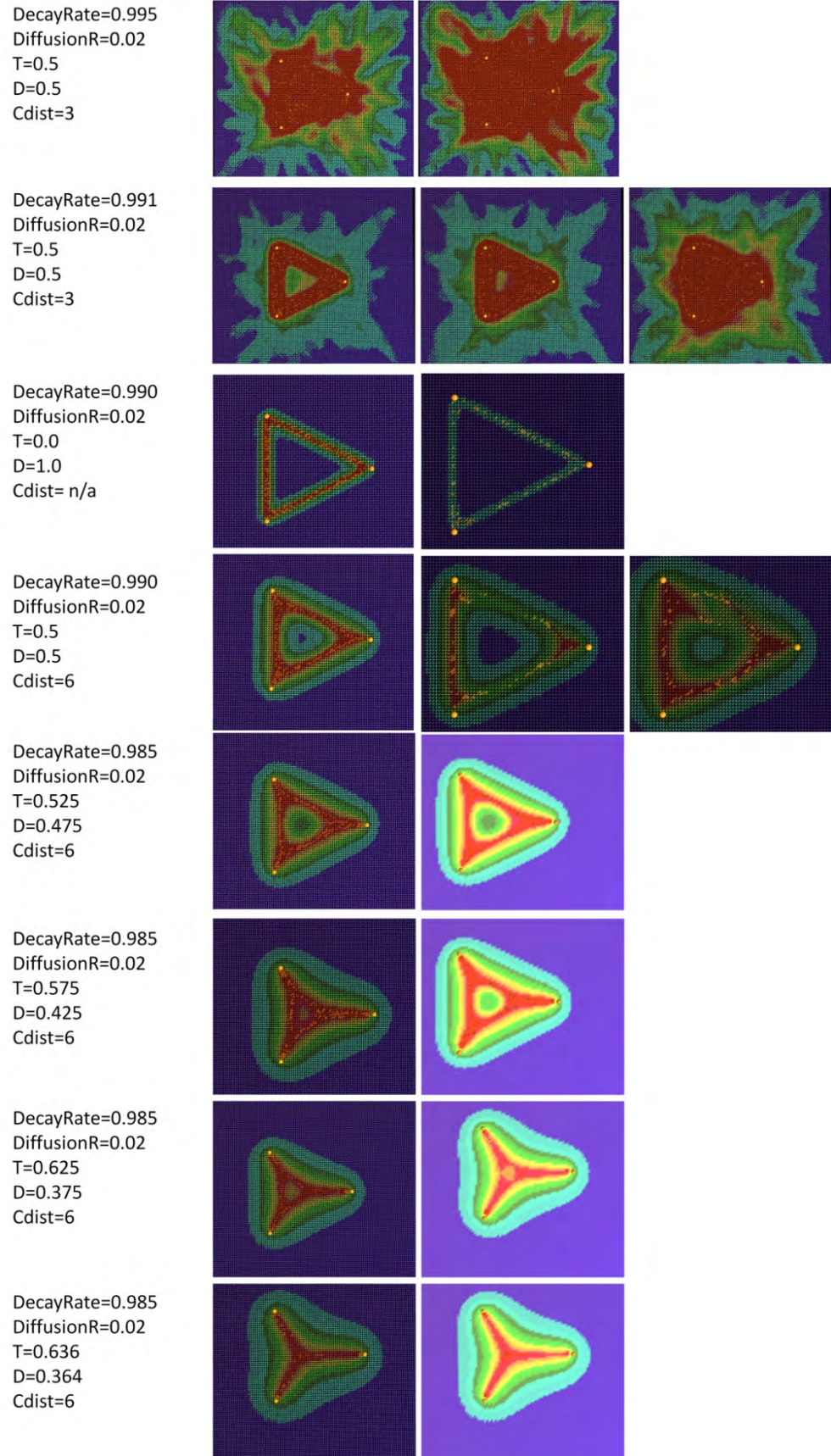


Figure 6.9b – Simulation on 3 Cities with various parameter settings.

Figure 6.9a and b shows the resulting configurations per different parameter settings. I used an agent population of 200 for this experiment. The chemical-rate for agents was set to 0.001, and the diffusion rate was fixed at 0.02. A decay rate of 0.995 caused saturation of traffic intensity in the system. The environment literally became all red due to high traffic intensity values at each cell, and agents' movements did not converge. This is because accumulation of chemicals by agents and diffusion of chemicals based on the rate exceeded the rate of evaporation of the chemical. Reducing the decay rate to 0.985 helped the system maintain traces of trails. (Otherwise all traces disappear if the decay rate is too low.)

1) Minimal Way Systems

Next, I started incrementing the t-factor from 0.0. In the case of $t=0.0$ and $d=1.0$, I observed a clear triangle forming, comprising the direct paths between the three vertices. Reducing the decay rate from 0.990 to 0.985 produced sharper edges on the triangle. When the t-factor value exceeded the d-factor value, the triangle started to deform, and paths that once met only at the vertices started to bundle together and merge. As a result, at $t=0.575$, the triangle started to shrink toward its center (see Figure 6.9). At $t=5.63$, all edges of the triangle merged together and formed into a three-legged starfish-like shape. More agents are attracted to frequently used trails, and agents gravitated toward the center of the triangle. This resulting starfish-like configuration forms a minimal way system (Schaur, 1991). A minimal way system provides a 57.7% shorter overall road length than a direct path system. (Note: This percentage refers to this specific setting.) In return, every agent must take about a 15.5% detour of added distances between cities

compared to a direct path. Schaur (1991) found that many existing road patterns exhibit a minimal way structure. An increase in traffic intensity factor (t-factor) becomes an incentive to form a minimal way system from a triangle-shaped direct path system. A shorter total distance of a system is economical in terms of construction costs of roads. Longer distances for each trip are compensated by shorter distance of entire length in a minimal way system.

In addition, the above results are all obtained when chemical distance is set at 6 grid units of the environment. (The size of the triangle formed by three cities has a 45-unit-length width and a 45-unit-length height.) The chemical distance governs an agent's maximum distance to detect traffic intensity. By reducing the chemical distance to 3, I could not obtain a minimal way system from any of the above parameter setting combinations listed in Figure 6.9.

It should be noted that these simulations of trail formation using multiple agents are a rather time-consuming process. The emergence of a minimal way path usually takes about 5 to 10 minutes of simulation time for the simple triangle configuration. As described later in this chapter, I conducted more complex simulations, and they required over an hour of simulation time. Agents' trails usually start from a direct path because there are no traffic intensity values embedded in the terrain at the outset. Agents gravitate toward more active areas of terrain and gradually form a new configuration such as a minimal way that corresponds to a programmed set of behaviors implemented on a group of agents.

Schaur (1991) has investigated empirical precedents of the above observable facts about a minimal way system. Schweitzer (1994) and Helbing et al. (2002) have done computational simulations using their “active walker model” and simulated the above-mentioned occurrences of a minimal way system in their papers. Helbing et al. noted that their model’s trail formations depend on one independent parameter in their dimensionless equation $k = IT/\sigma^2$ where I represents the intensity of footprints by pedestrians, T the durability of trails, and σ the visibility of a place.

This result implies that the emergence of street networks among humans is not all motivated by efficiency based on travel distances. Although a minimal way system is not as efficient as a direct way system, it provides a different connectivity and topology of the networks. A mid-point of star-fish shape becomes a convenient point for connections between three other cities and promotes emergence of another city (population concentration area) at the location. If travelers would like to maintain options for their travel routes until they reach the midpoint, this is a good layout, too. Later, I implement emergence of cities based on traffic intensity, and this is a good example of a new city emergence pattern often seen in many human systems.



Figure 6.10 – Emerging Trail Patterns

6.2.5 S + T + D Model

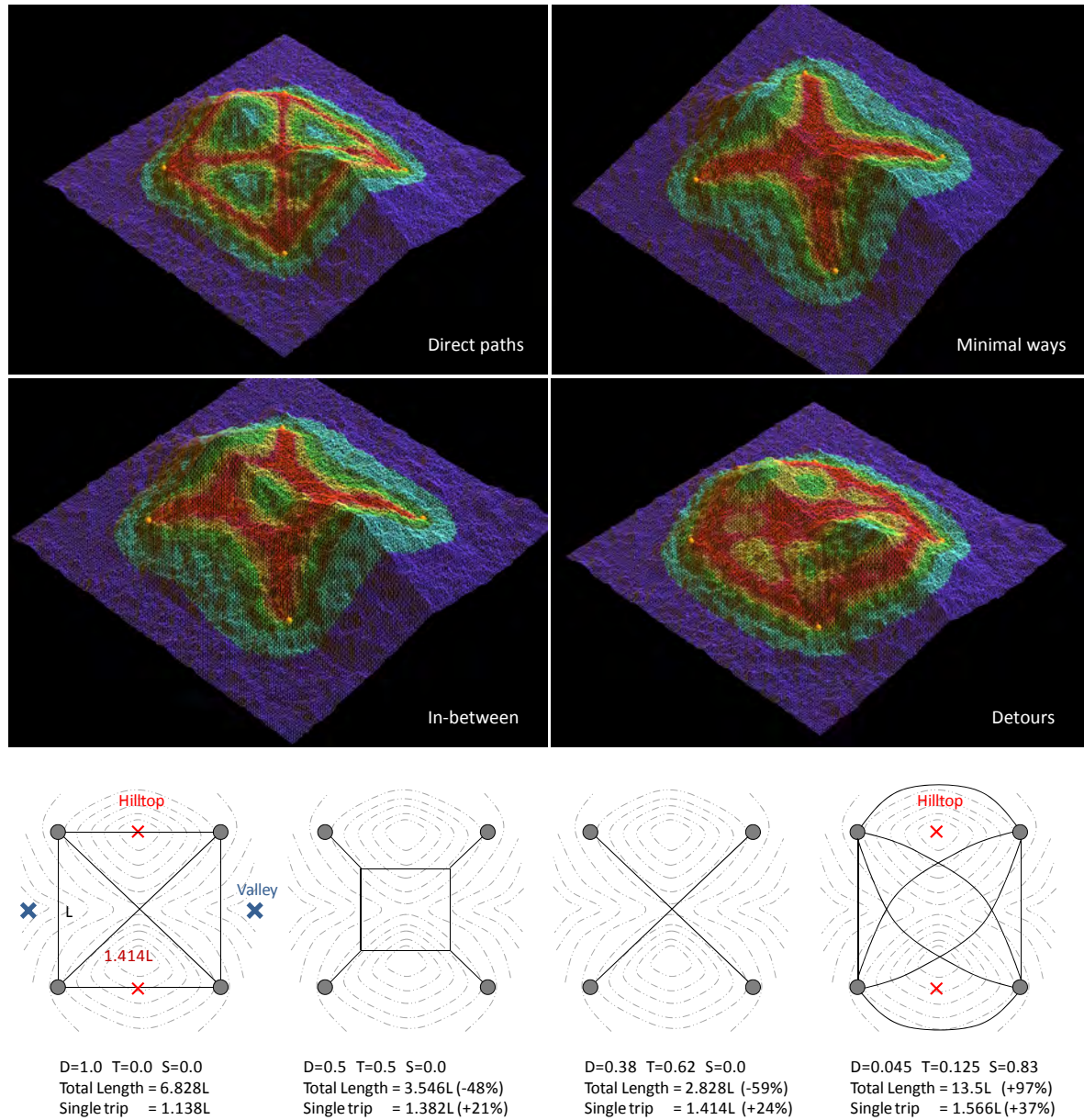


Figure 6.11 – 4 Cities on uneven terrain: Transition among direct paths, minimal ways, & detours

In this section, I would like to integrate the concept of slope-factor into the above T + D model. Models using attraction to traffic intensity and destinations have been studied by several scholars; however, to my knowledge, a model implementing pedestrian agents' attraction to gentle slopes has not been explored.

For this model, four cities are defined at vertices of a square in a plan view, and agents will tour around these cities. The major difference from the last T + D model is that the environment is located on an uneven terrain. There are hills on the top and bottom sides of the square and valleys on the left and right sides of the square. The path between any two cities thus requires agents to encounter a hill or valley. As I described earlier in this chapter, agents' heading vector, V_{slope} , is defined by slope and always tries to maintain a gentler slope direction for agents, and s-factor is the weight that defines a proportion of this directive force vector relative to the two other vectors, $V_{traffic}$ and V_{dest} affected by their own factors t and d . Derivation of V_{slope} involves two parameters, maximum-slope-angle and slope-distance. Simulations are carried out using various proportions for the three factors, s , t , and d , and two slope-related parameters.

The results were the same while s-factor stayed at 0.0, and transition between a direct path and a minimal way was observed as I increased the proportion of t-factor relative to d-factor (see Figure 12). If s-factor is large, agents start to choose more comfortable paths by avoiding peaks and beginning to detour. In the case of $d=0.045$, $t=0.125$, and $s=0.83$, agents produced detours around the two peaks by maintaining altitudes in a relatively narrow range. Use of the s-factor has become an incentive for agents to produce paths along contour lines on surfaces. Use of a high s-factor also diminishes attraction to destinations, and trails tend to diffuse and get wider as a result. In the case of $d=0.38$, $t=0.62$, and $s=0.0$, a minimal way system emerges. If the d and t factors stay the same and the s-factor increases, a compromise results between ease of walking and overall path distance.

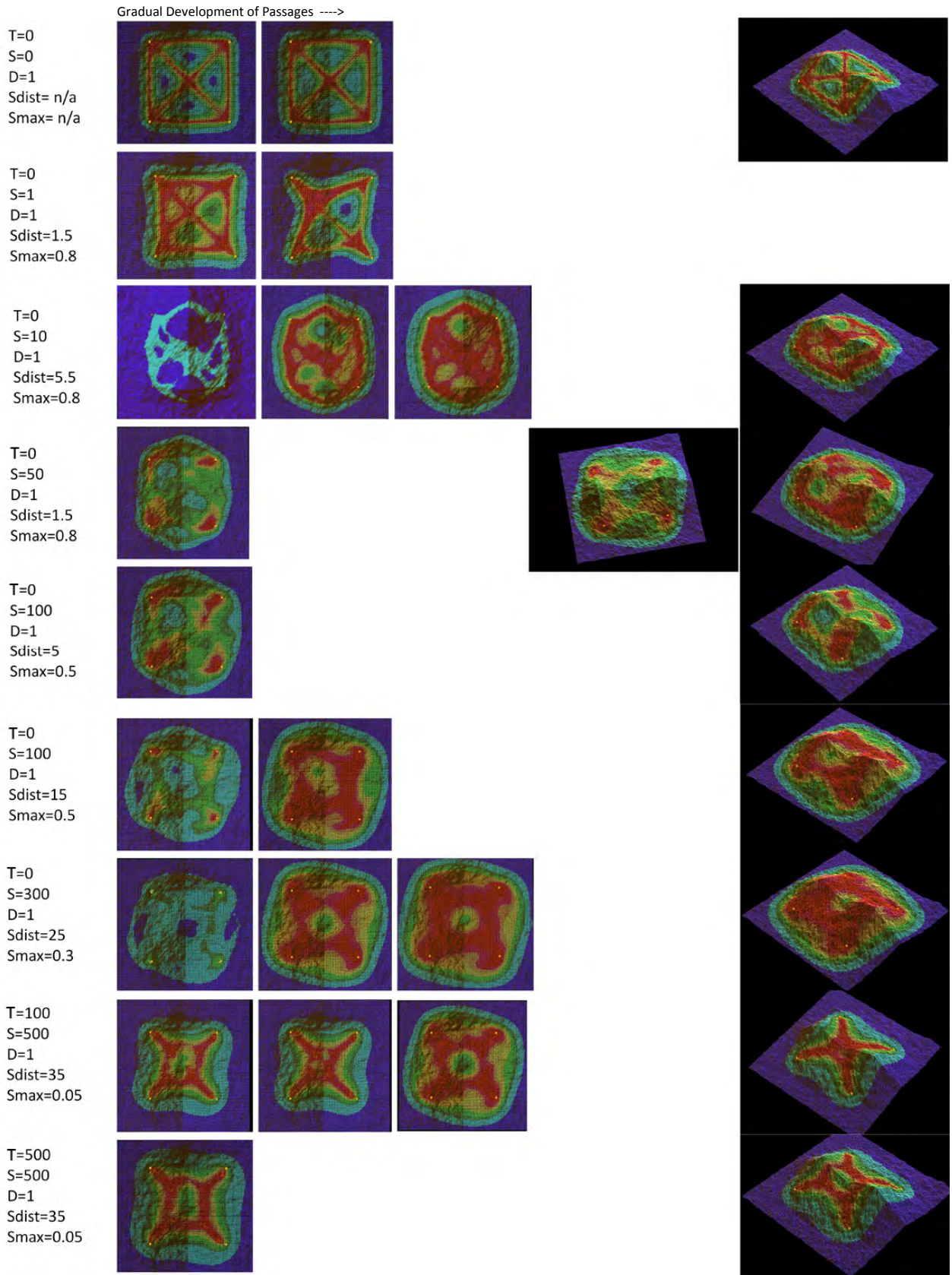


Figure 6.12a – 4 Cities on uneven terrain: Results with various values for S, T, and D factors. D is set as a constant value 1.0 above. (Actual D value in percentage = $D/(D+S+T)*100$)

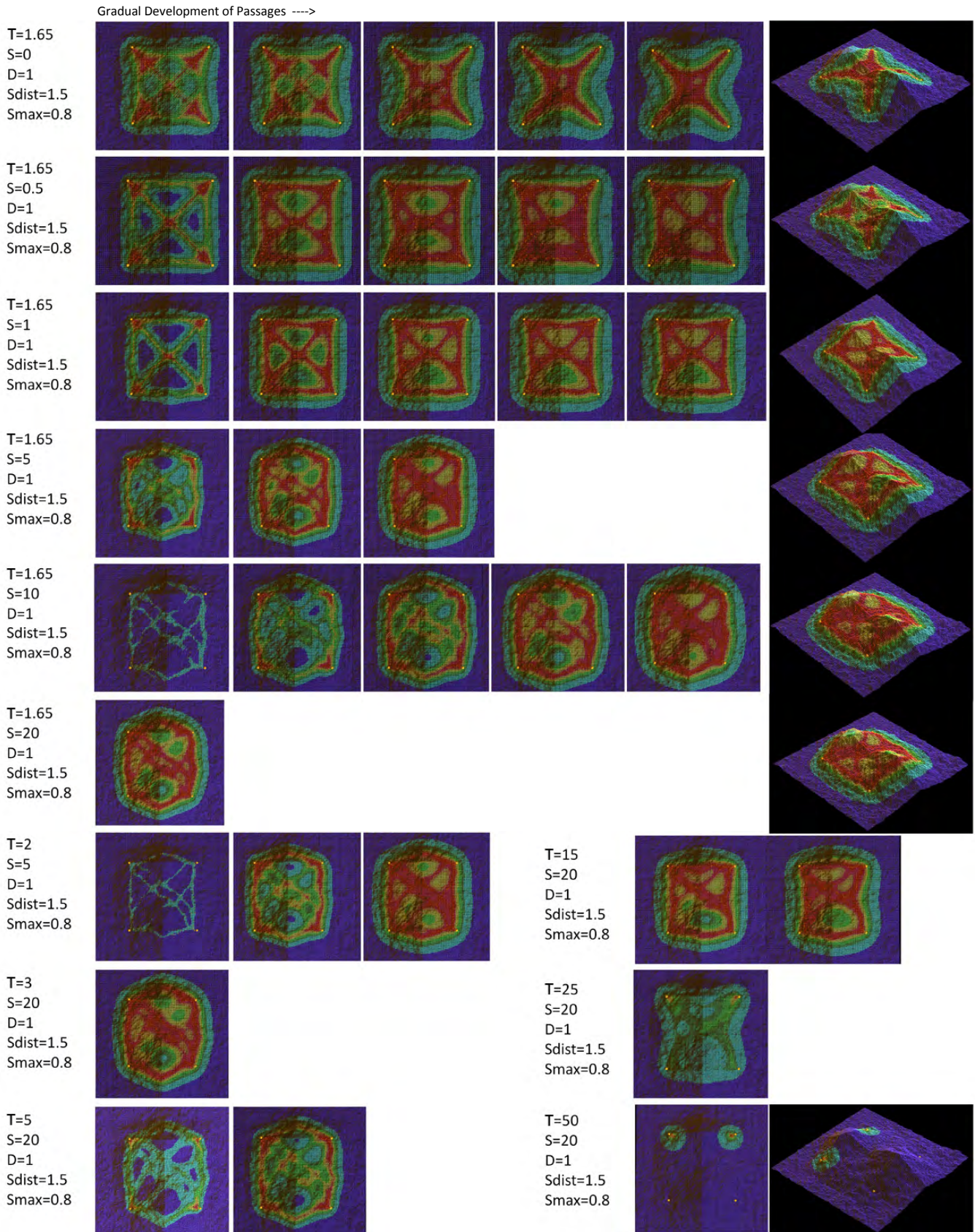


Figure 6.12b – 4 Cities on uneven terrain: Results with various values for S, T, and D factors. D is set as a constant value 1.0 in these cases. (Actual D value in percentage = $D/(D+S+T)*100$)

Instead of using the regular four corner points on a square, four arbitrary points on rough and hilly terrain are selected as cities in the next simulation (Figure 14). Use of arbitrary points makes simulation results more unpredictable. However, I gained the same interrelationships between three configurations (direct path, minimal way, and detour) and three factors for agents' behaviors (t, s, and d). Later, I increased the number of cities at arbitrary locations to 8 and 9. Figures 14a to 17d show the resulting configurations.

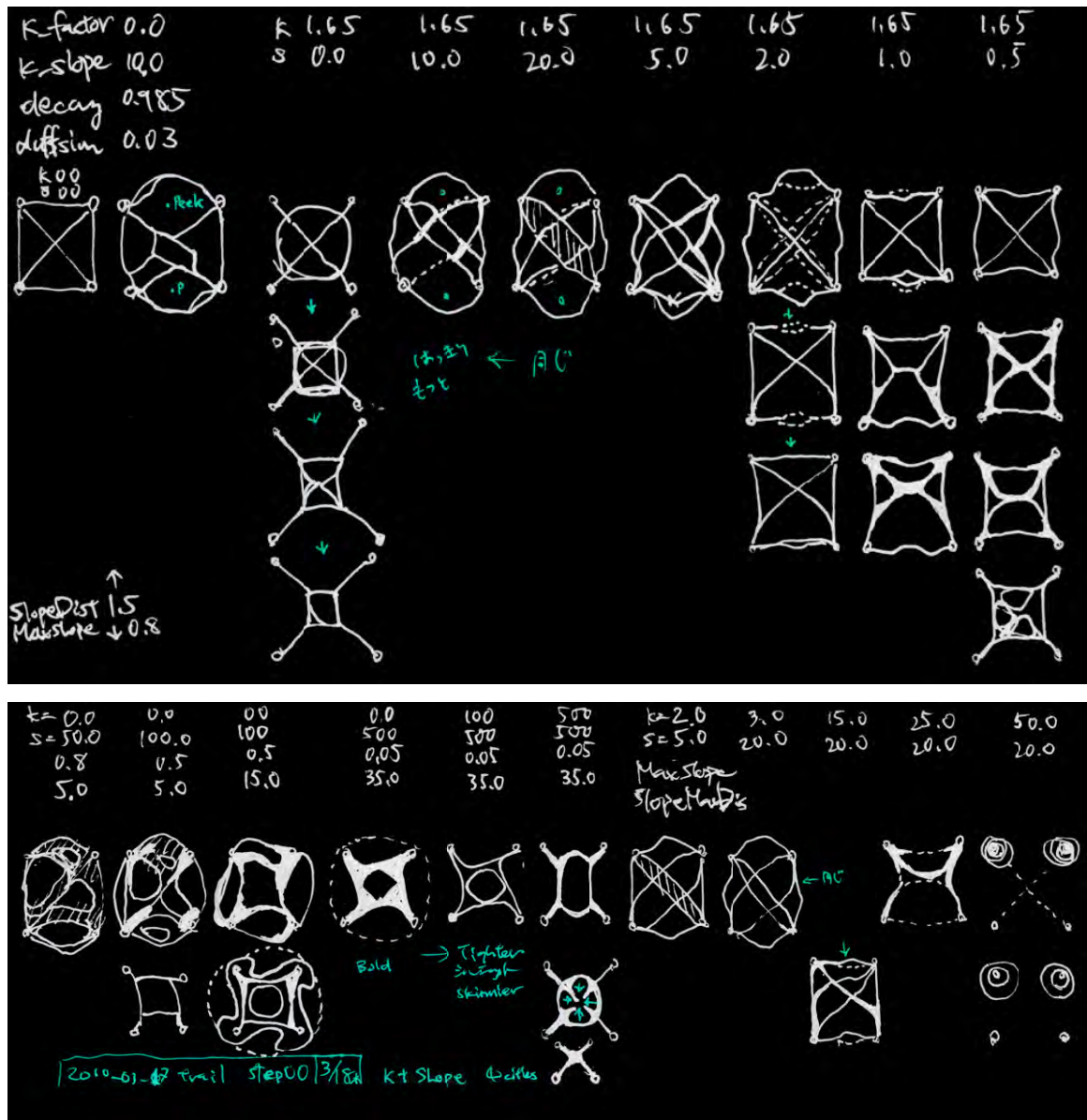


Figure 6.13 – Topological representations of 4 cities with various parameter settings.

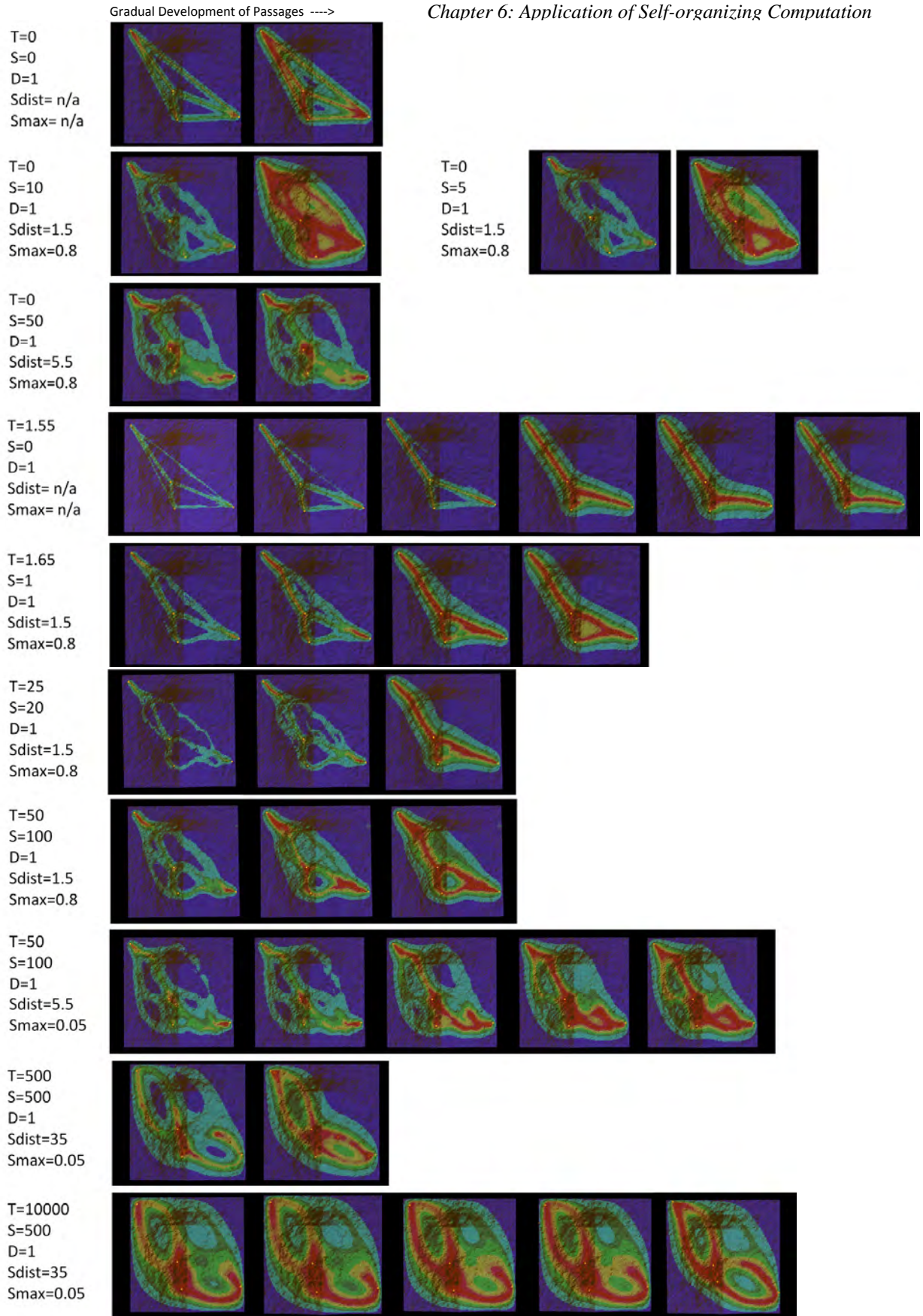
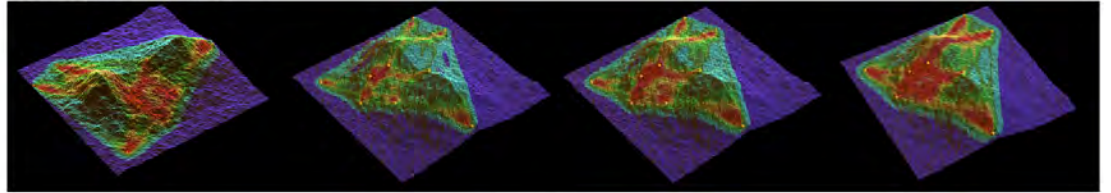


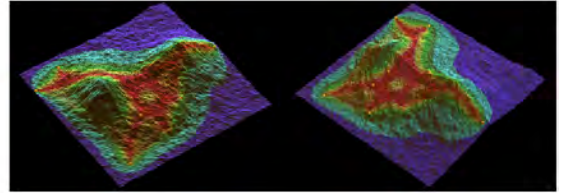
Figure 6.14 – 4 cities at arbitrary locations on uneven terrain with various S, T, D factors. D is set as a constant value 1.0 above. (Actual D value in percentage = $D/(D+S+T)*100$)

T=0
S=0
Sdist= n/a
Smax= n/a

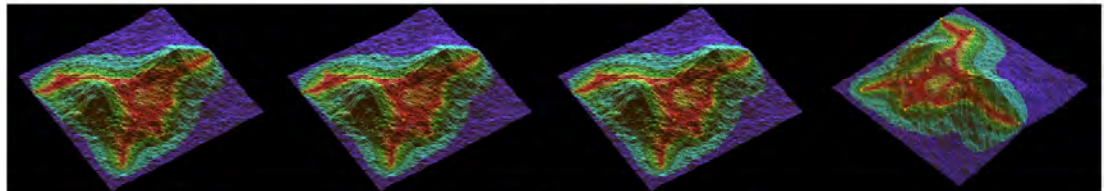
Gradual Development of Passages, ---->



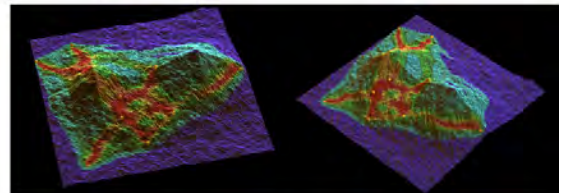
T=0
S=500
Sdist=35
Smax=0.05



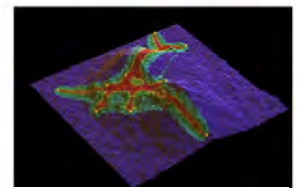
T=0
S=2500
Sdist=35
Smax=0.05



T=1.65
S=5
Sdist=1.5
Smax=0.8

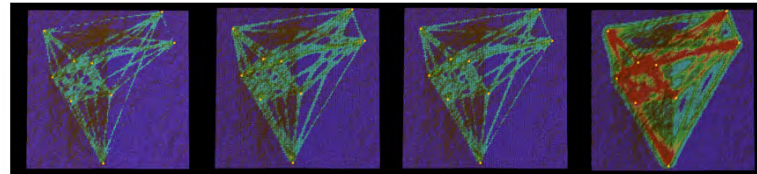


T=1.65
S=0
Sdist= n/a
Smax= n/a

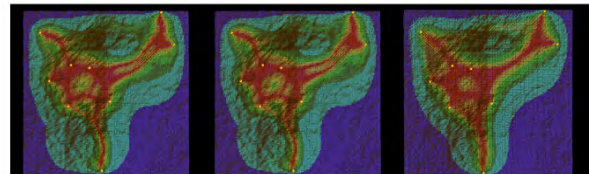


Gradual Development of Passages ---->

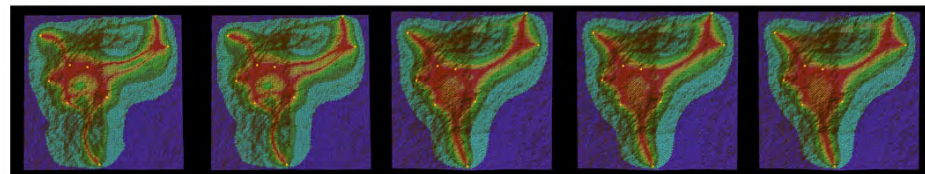
T=0
S=0
Sdist= n/a
Smax= n/a



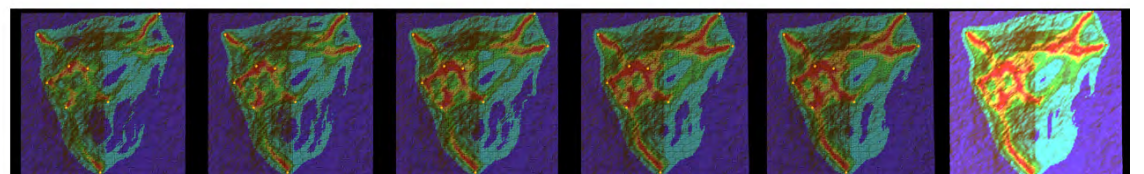
T=0
S=500
Sdist=35
Smax=0.05



T=0
S=2500
Sdist=35
Smax=0.05



T=1.65
S=5
Sdist=1.5
Smax=0.8



T=1.65
S=0
Sdist= n/a
Smax= n/a

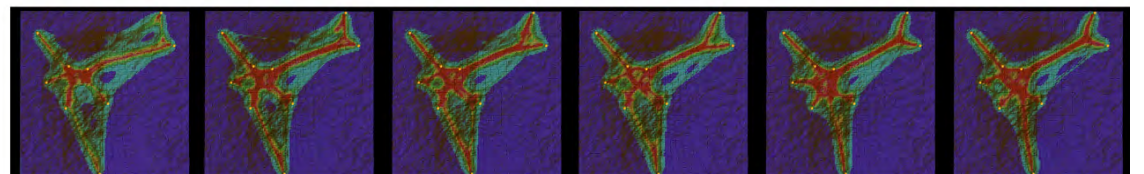


Figure 6.15a – 9 cities on Arbitrary locations on uneven terrain with various S, T, D factors.
D is set as a constant value 1.0 above. (Actual D value in percentage = $D/(D+S+T)*100$)

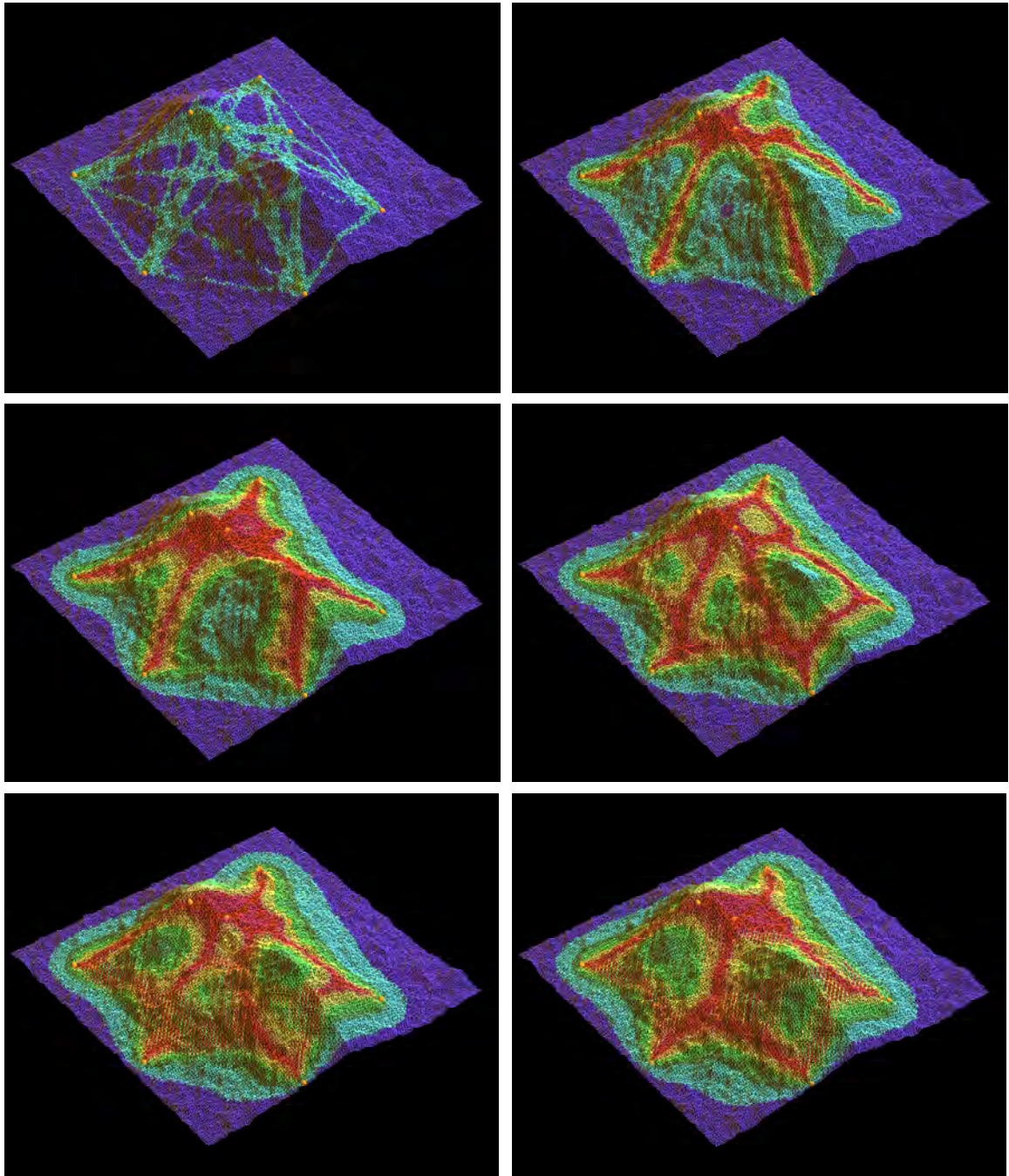
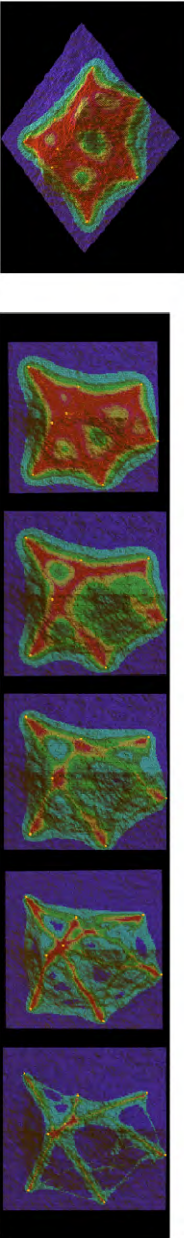


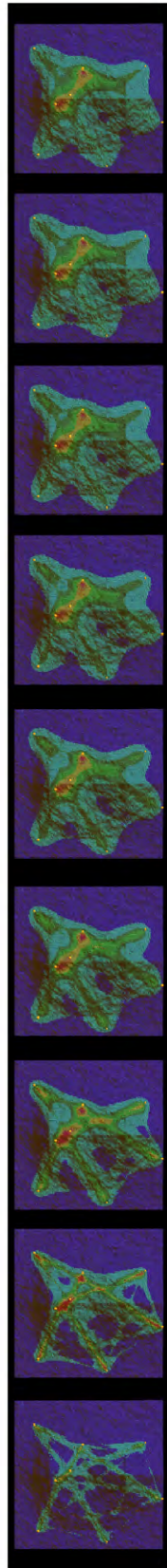
Figure 6.16 – 8 cities with various parameter settings displaying different topologies.
Figure 6.17a – d: – show gradual growth sequences based on different parameter settings (next pages)

8 Cities: Gradual Growth Processes of Street Networks.

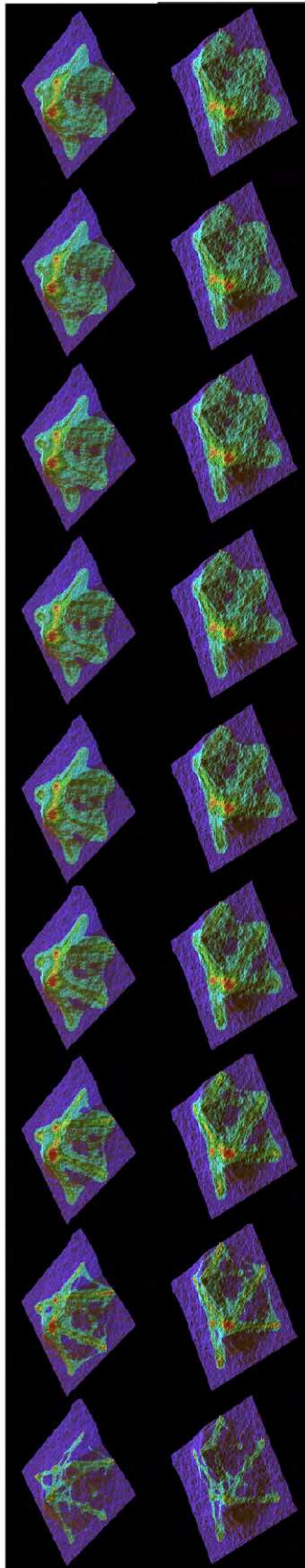
Gradual Development of Passages. ----->



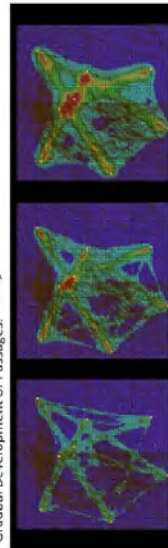
T=1.65
S=1
D=1
Sdist=1.5
Smax=0.8



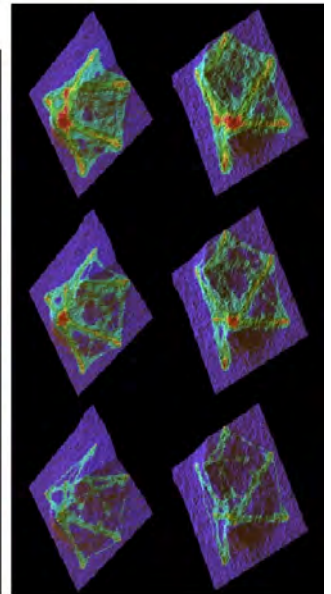
T=1.65
S=0
D=1
Sdist= n/a
Smax= n/a



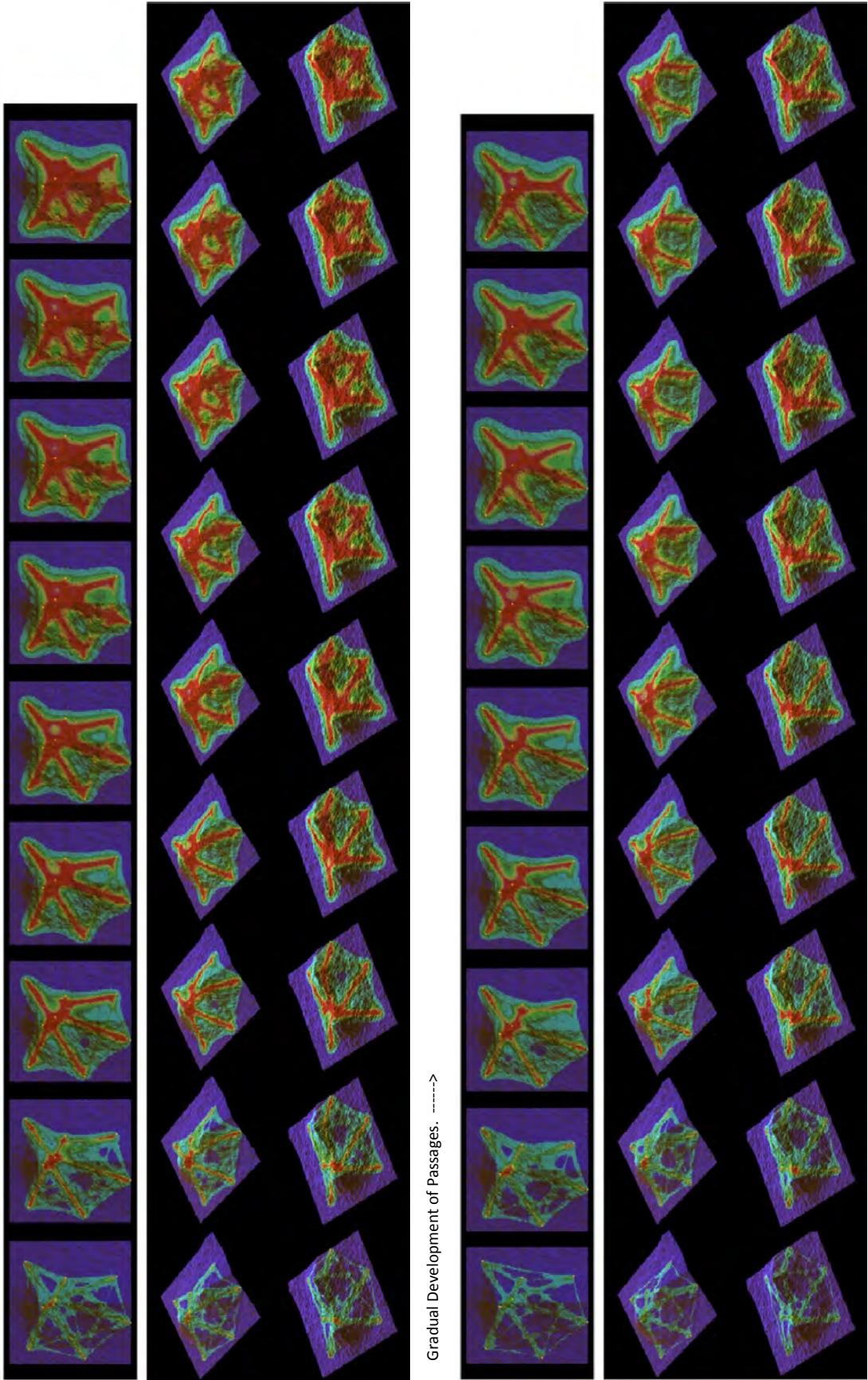
Gradual Development of Passages. ----->



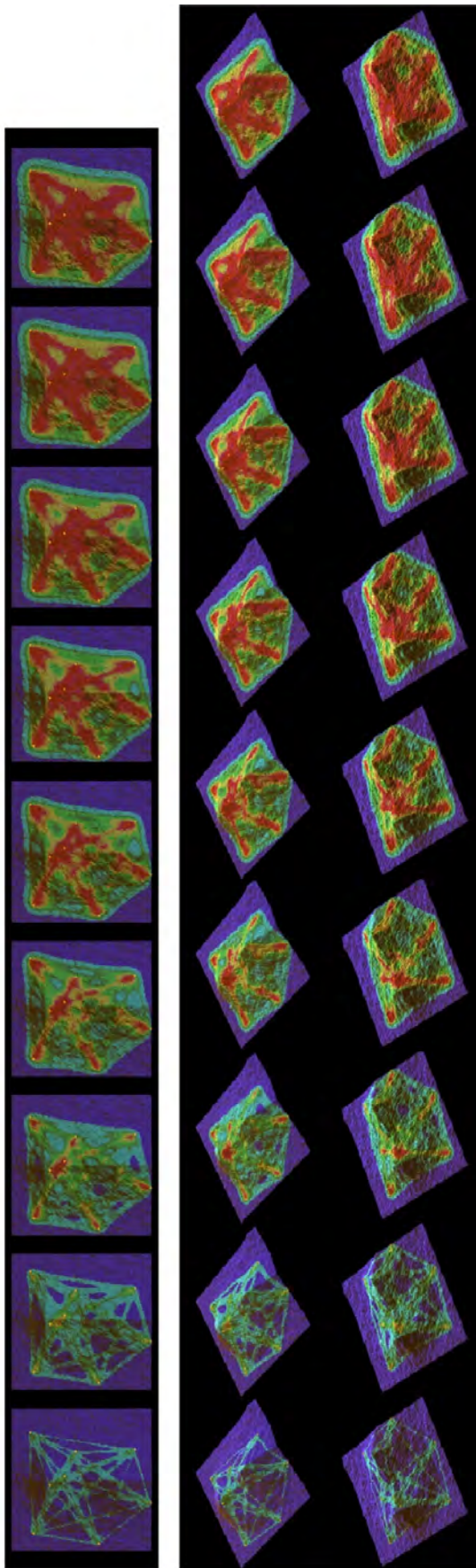
T=1.55
S=0
D=1
Sdist= n/a
Smax= n/a



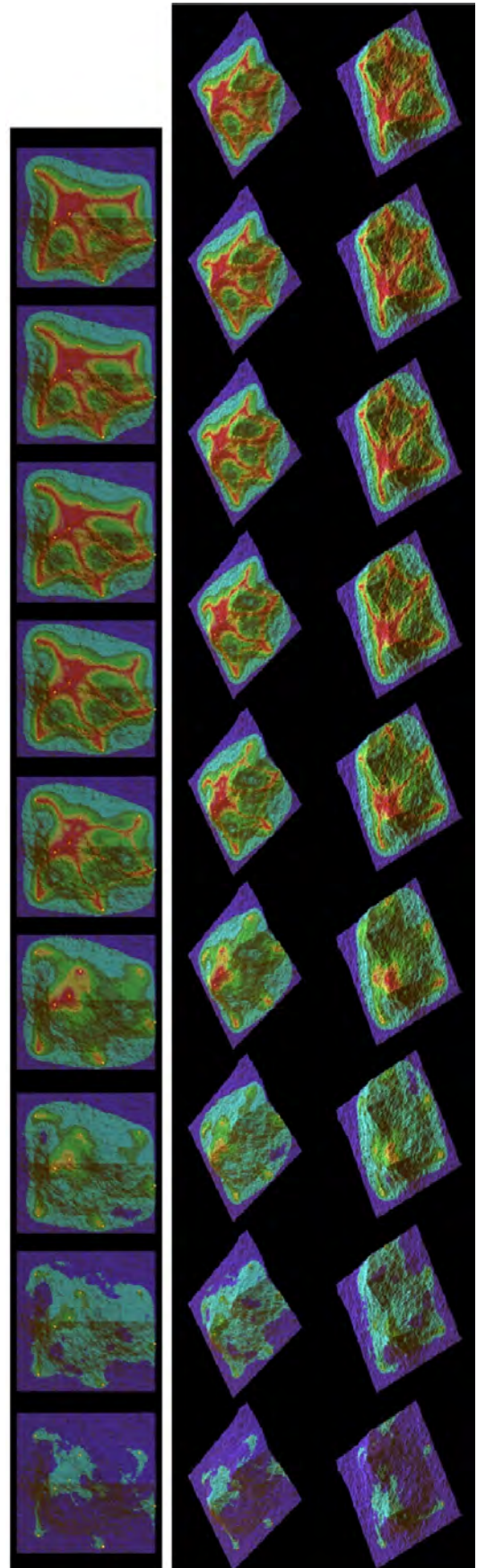
8 Cities: Gradual Growth Processes of Street Networks.



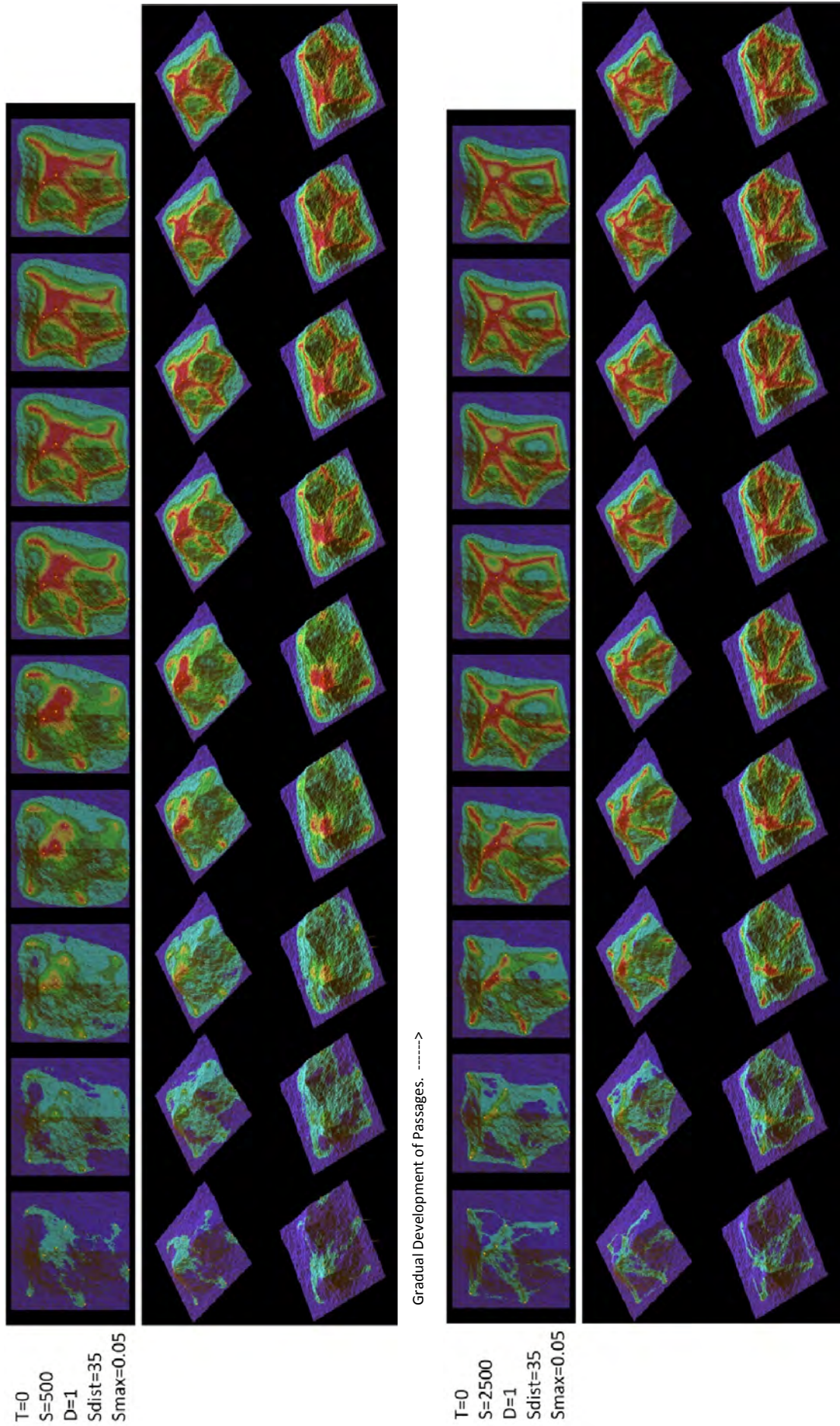
8 Cities: Gradual Growth Processes of Street Networks.



Gradual Development of Passages. ----->



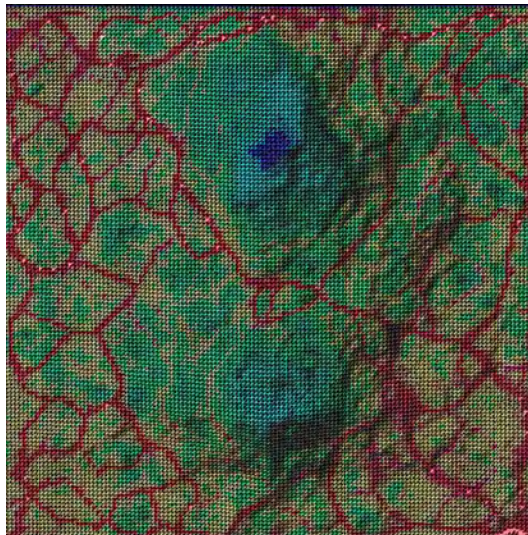
8 Cities: Gradual Growth Processes of Street Networks.



Summary of the Results

The high value for s-factor (slope) induces agents to find a comfortable walking path. By avoiding climbing or descending a steep hill, agents produce detours. The high value for d-factor (destination) induces agents to find the shortest path. As a result, agents produce a direct path system. The high value for traffic-intensity-factor stimulates agents to minimize overall length of circulation. When agents have more frequent trips between destinations, shorter overall length of the system is beneficial due to its lower construction costs of roads. These three factors can be applied in various different proportions to find compromise solutions among three different motivations. One of the biggest merits of the system is that purely microscopic behaviors can produce a global configuration without requiring any macroscopic information to be input into the system.

When numbers of cities are small, resulting configurations are rather predictable. However, deriving a street configuration from a large number of cities at arbitrary locations on irregular terrain geometry is a challenge. Multi-agent simulation is not necessarily the fastest method of derivation, but it is a reasonably robust method for deriving street configuration as it only requires microscopic behaviors as input information.



**Figure 6.18 –
Emergent Trails
(from next section)**

6.2.6 Emergent Behaviors of Agents

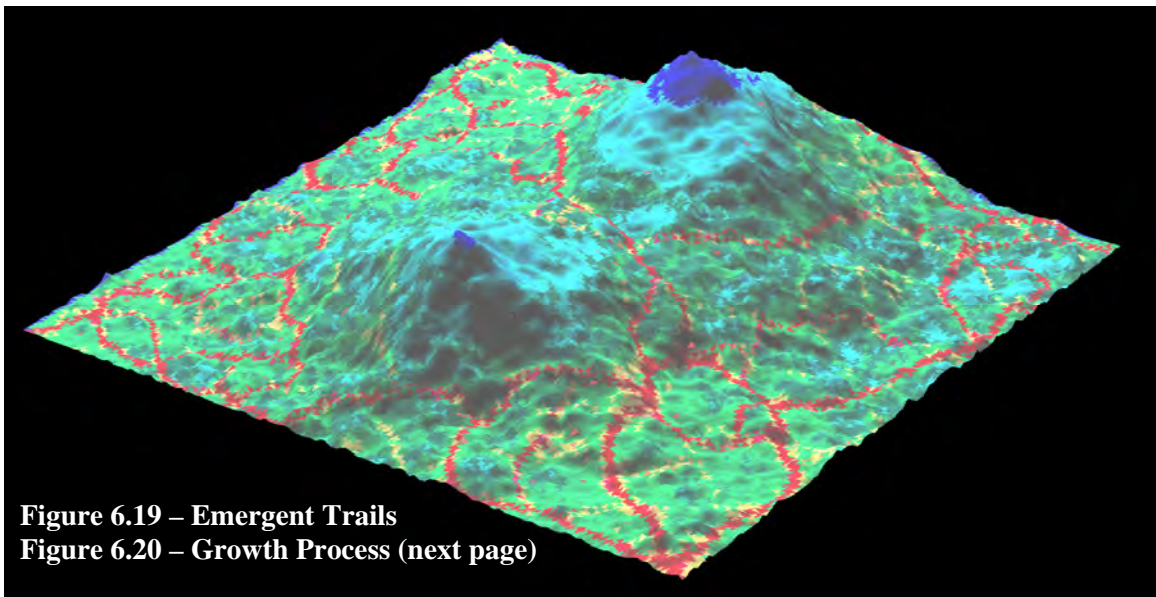
Experiments in the last section have fixed agents' behaviors throughout the simulations. In the following section, I would like to consider shifts in agents' behaviors stimulated by environmental changes.

1) Early Periods

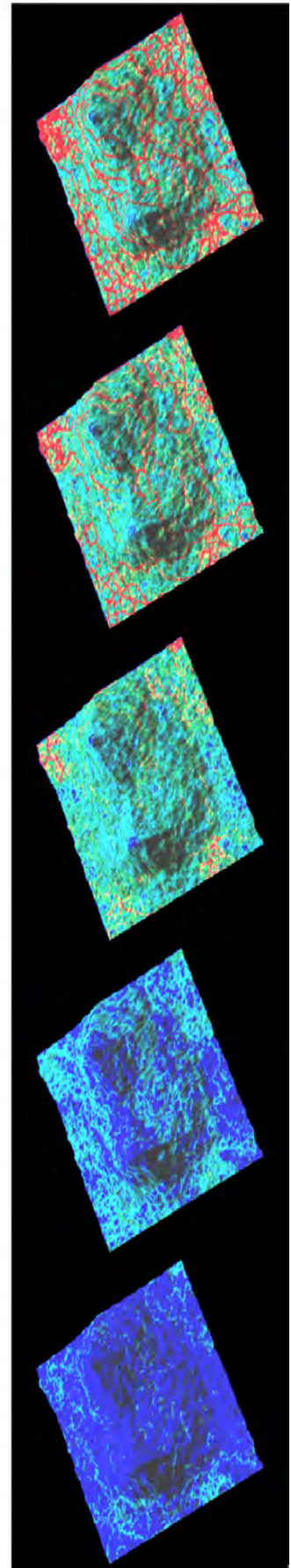
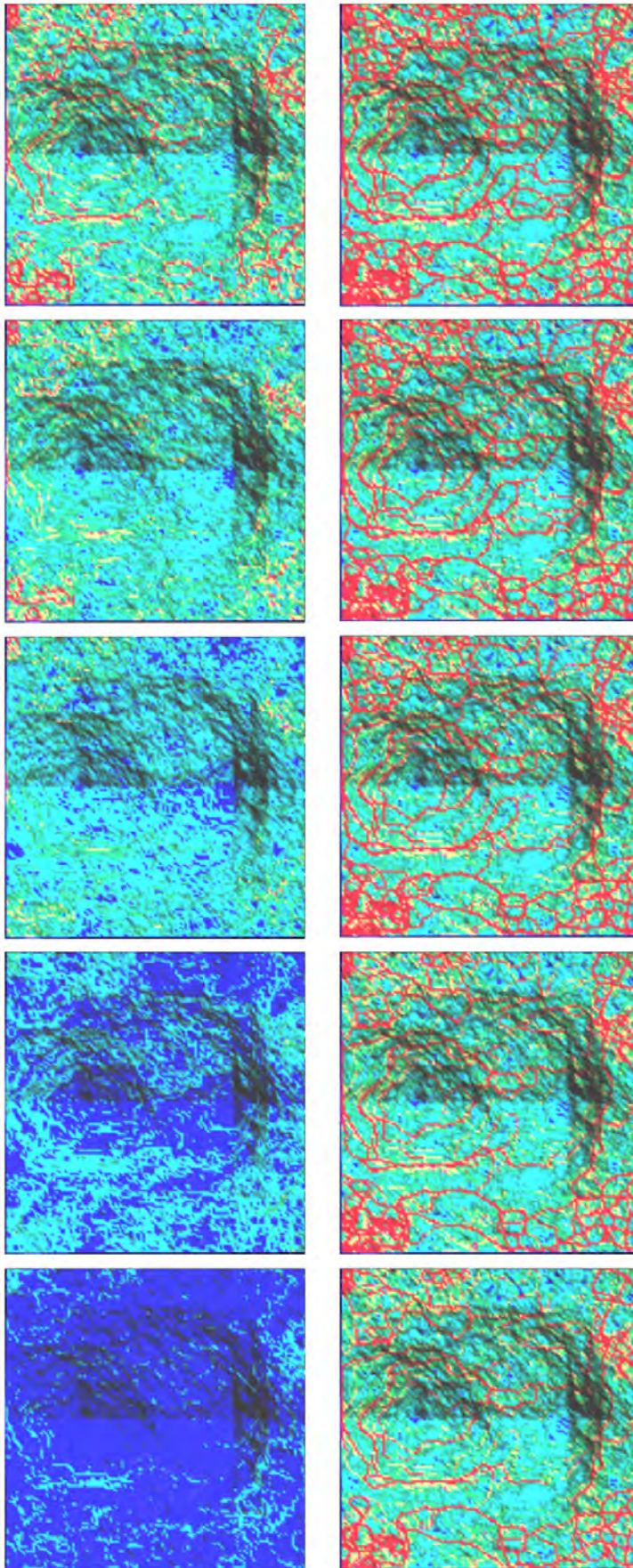
Firstly, I focused on slope-factor and traffic-intensity-factor. I chose a randomly generated terrain by mid-point displacement method and started the simulation with 200 agents with only attraction to gentler slopes. However, I set the agents' behaviors to shift after the entire terrain's total traffic-intensity value exceeds a certain minimum. An original state of the site is assumed to have no trails and no trace of human settlement. Once an entire environment's activity level reaches a certain maturity, agents start to rely on information that already exists in environments. Agents start to follow higher traffic frequency areas instead of trails that no one has been taking. From the original part of the simulation, agents have already left better pathways in terms of terrain slope, and by selecting more frequently used paths, agents can avoid steep and undesirable paths. This phase change in agents' behaviors motivates emergence of sharply outlined passage ways on the terrain. I tested the results with different values for agents' extended vision for the slope (slope-distance from 1.0 to 5.0 unit distance of the environment). Figure 20 shows gradual appearance of street networks. The terrain has two steep hilltops at the south and north, and detours around these hilltops are recognized from the results. The results are highly influenced by the slope-distance value. A finer and more intertwined network of passage ways covering higher regions of hills is recognized as I increase the slope-

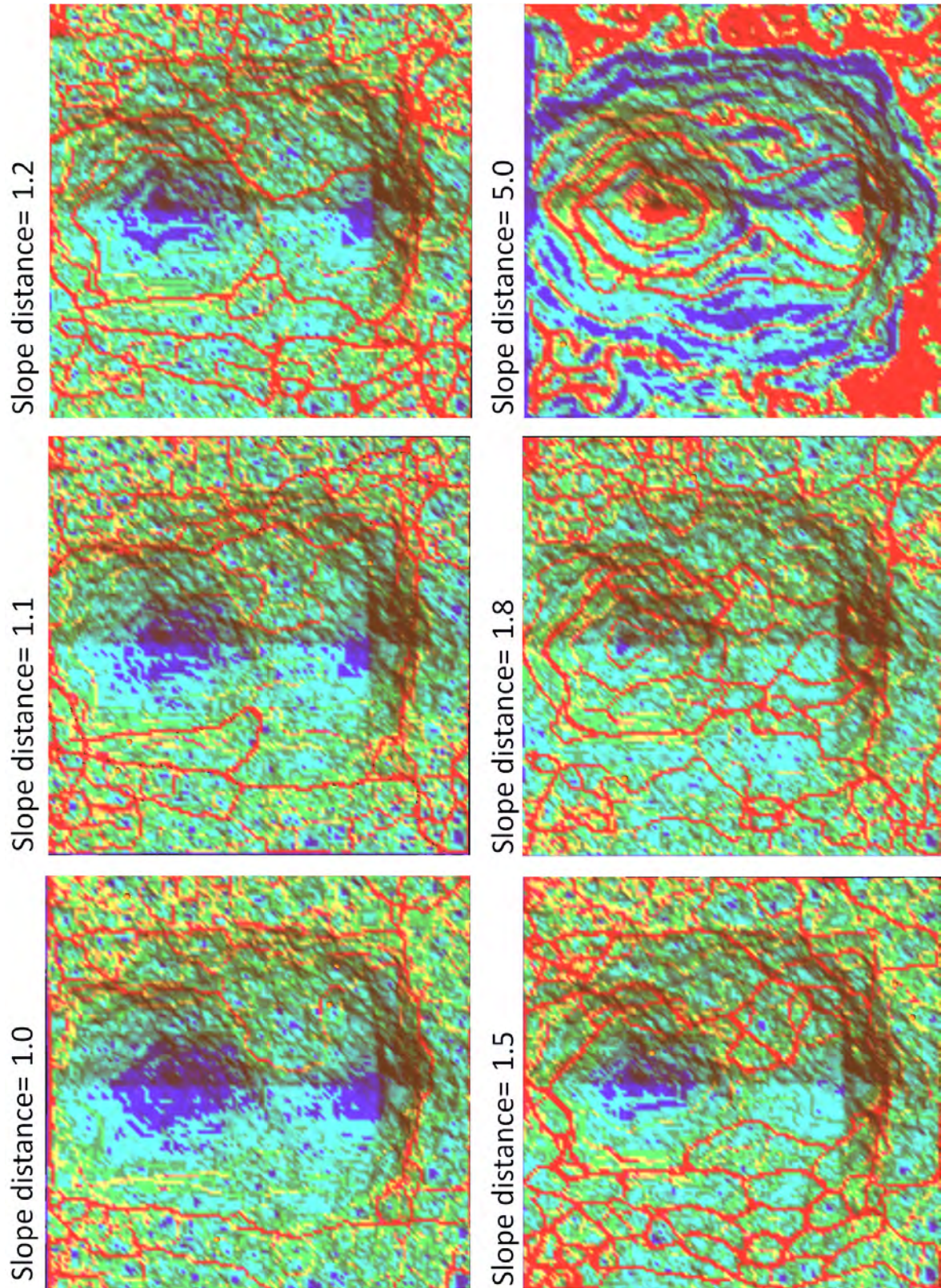
distance value. Some of the passage ways have consistently appeared regardless of the slope-distance value, and these indicate the emergence of possible artery systems for the region.

The threshold value that triggers the agents' behavioral phase change was carefully chosen. If the value is too low, the entire site's recorded frequencies of traffics are too premature and too biased by a seed value of a random generator that defines agents' ambient movements. As the number of terrain patches that possess traffic intensity value of over 0.8 exceed 15% of the total number of patches, agents start to check the traffic frequency around them to make decisions about their heading directions. (Chemical-rate of agents is set to 0.001. In order to reach 0.8, agents need to pass this patch 800 times.) In the case of this specific example of terrain with two humps, the above condition was satisfied when an average intensity value for patches reached 0.15. Instead of looking at a pure average value as a threshold, I used the proportion of eccentric conditions in order to execute the behavioral change. This decision was empirically made based on results from many different threshold values. Even if the overall average value becomes high, this fact does not always guarantee the maturity of data from agents' movements.



Slope Distance = 1.5 unit
Maximum Tolerable Slope = 0.8





Figures 6.21 – Gradual Emergence of Trails (previous page); Results with various slope-distance parameters for agents.

2) City Emergence

After the emergence of street networks, agents' movements become much more ordered and organized. Some of the intersections of several arteries become population concentration areas, and these areas have the potential to grow into cities. As the number of terrain patches that possess traffic intensity value of over 0.8 exceed 18% of the total number of patches, the system starts to check for and find traffic-intensive areas. This is the same threshold that triggers agents' behavioral shifts. However, this new action occurs after the emergence of street networks. The algorithm finds peak areas of traffic intensity value above a certain threshold value (0.9 was used), and finds places where these peaks are forming clusters. Traffic-intensive patches that are within a certain distance from each other are read as one island. If these islands are larger than a certain minimum size, they are considered as city areas. Within these city areas, the highest traffic intensity peak location becomes a city center. If a city area has more than one peak and if they are beyond a certain distance apart from each other, the city area can have more than one city center. These values – a minimum size of an island and a minimum distance between city centers – are parameters that users need to define. After sufficient careful trials, numbers that produce results that represent a natural scale of development relative to the scale of terrain are empirically adopted. The scale of the system and the scale of the input information is a critical issue that I would like to discuss again later in this chapter.

```

Procedure city-Emerge( ){
    // check if environment is mature enough.
    If (Total intensity of traffic exceeds a minimum threshold)

        // extract peaks
        Peaks[] = SearchPeaks();

```

```

// create sub-cities by grouping peaks within Min-distance
foreach( peak1 & peak2 in Peaks[] ){
    If( distance(peak1, peak2)< Min-Dist)
        Subcities[[]].add(peak1, peak2);
}
// select sub-cities with sizes over Min-Number-of-City
foreach( Subcities[] ){
    If(Subcities[i] > MinNumCity)
        Cities.add(Subcities[i]);
}
// extract city center points from Cities
foreach( Cities[] ){
    //
    for( Cities[i].count )
        If(Cities[i] has only 1 peak)
            Destinations.add(Cities[i].peak);
        If(Cities[i] has more than 1 peaks)
            foreach(j in Cities[i].peaks)
                If(peaks are MinDistCity away){
                    Destinations.add(Cities[i].peaks[j]);
                }
            Else(peaks are all within MinDistCity){
                pkBest=(select highest-peak from peaks);
                Destinations.add(pkBest);
            }
        }
    }
Return Destinations[];
}
}

```

The canonical algorithm for the city center points extraction

3) Itinerary for Agents

Once cities are registered by the system, all agents will know about the locations of the cities. Agents have one destination city at a time, and they head toward that direction. This information about cities becomes global knowledge for agents. Unlike other decision-making factors for agents, such as slope angles and traffic intensity, these are globally defined relatively static information that affects agents' behaviors; however, these cities can become extinct or re-emerge from other locations. Over the course of simulation, the system can recheck the traffic concentration areas to update information. Once agents reach their current destinations, they will be assigned new destinations

randomly selected from the list of cities (excluding the most recent destination). If all agents simply head toward their destinations, a direct path system emerges.

At the time of the emergence of the cities, the environment already has developed networks of trails induced by agents' behaviors. Even if there are newly defined destinations, agents' behaviors should be influenced by these existing trails. Some trails may be roundabout in terms of distances to the destinations, but they may provide more comfortable walking experiences. Establishing a newly defined pathway is also a cumbersome task. As I mentioned earlier, agents' heading directions can be controlled by assigning different proportions of weights for each vector toward different attractors – gentler slopes, traffic intensities, and destinations. Original trails before city emergence are similar to a mountain trail as they are generated based on the relatively myopic views of agents at the early stages of development of the organization as a whole. Locations of the destinations are factors that emerged as a result of preceding developments; however, their influence can impose more directly upon agents and their environments.

Different proportions of values for three factors – slopes, traffic intensity, and destinations – were applied and various growth patterns of passage networks were recorded. In case of $S=0$, $T=0$, $D=1.0$, a direct path connecting all the destinations appeared. This is a relatively trivial result; however the gradual transformation from mountain-trail-like complex paths to straight roads is quite dramatic.

See Fig.6.23-A

```
//DIRECT PATH
(T=0%; S=0%; D=100%)
Agent0.T_factor      = 0;
Agent0.Slope_factor  = 0;
Agent0.ChemRate       = 0.012f;
Agent0.TransMoveS     = 1.0f;
Agent0.slopeDistMax   = 1.5;
Agent0.MaxSlope       = 0.8;
Agent0.ChemMaxdist    = 6;
Primitive.decayRate   = 0.9995f;
Primitive.diffusionRate = 0;
```

For the following cases, I recorded hybrids between a minimal way system and a detour.

See Fig.6.23-B

```
//GOOD
(T=58%; S=0%; D=42%)
Agent0.T_factor      = 1.35;
Agent0.Slope_factor  = 0;
Agent0.ChemRate       = 0.012f;
Agent0.TransMoveS     = 1.0f;
Agent0.slopeDistMax   = 1.5;
Agent0.MaxSlope       = 0.8;
Agent0.ChemMaxdist    = 6;
Primitive.decayRate   = 0.9995f;
Primitive.diffusionRate = 0.0f;
```

See Fig.6.23-C

```
//GOOD
(T=46%; S=27%; D=27%)
Agent0.T_factor      = 1.65;
Agent0.Slope_factor  = 1;
Agent0.ChemRate       = 0.012f;
Agent0.TransMoveS     = 1.0f;
Agent0.slopeDistMax   = 1.5;
Agent0.MaxSlope       = 0.8;
Agent0.ChemMaxdist    = 6;
Primitive.decayRate   = 0.9995f;
Primitive.diffusionRate = 0.0f;
```

In the rest of the cases, balances among parameters – slope-factor, traffic-intensity-factor, chemical-rate, decay-rate, and diffusion-rate – have become quite sensitive. For example, the following parameter combinations cause saturation of traffic intensity inside the environments, and literally turn all terrain red. This is mainly due to the balance between decay-rate and diffusion-rate. These parameters are important for the generation of a minimal way system.

See Fig.6.23-E

```
//TOO FAT RED
(T=10%; S=75%; D=15%)
Agent0.T_factor      = 0.65;
Agent0.Slope_factor  = 5;
Agent0.ChemRate       = 0.012f;
Agent0.TransMoveS     = 1.0f;
Agent0.slopeDistMax   = 1.5;
Agent0.MaxSlope       = 0.8;
Agent0.ChemMaxdist    = 6;
Primitive.decayRate   = 0.9995f;
Primitive.diffusionRate = 0.001f;
```

```
//TOO FAT RED
(T=27%; S=42%; D=35%)
Agent0.T_factor      = 0.65;
Agent0.Slope_factor  = 1.2;
Agent0.ChemRate       = 0.012f;
Agent0.TransMoveS     = 1.0f;
Agent0.slopeDistMax   = 1.5;
Agent0.MaxSlope       = 0.8;
Agent0.ChemMaxdist    = 6;
Primitive.decayRate   = 0.9995f;
Primitive.diffusionRate = 0.001f;
```

In the following cases, decay-rate was too fast compared to diffusion-rate, and eventually all trails disappeared.

```
//BAD ALL BLUE
(T=43%; S=31%; D=26%)
Agent0.T_factor      = 1.65;
Agent0.Slope_factor  = 1.2;
Agent0.ChemRate      = 0.012f;
Agent0.TransMoveS    = 1.0f;
Agent0.slopeDistMax  = 1.5;
Agent0.MaxSlope      = 0.8;
Agent0.ChemMaxdist   = 6;
Primitive.decayRate   = 0.985f;
Primitive.diffusionRate = 0.002f;
```

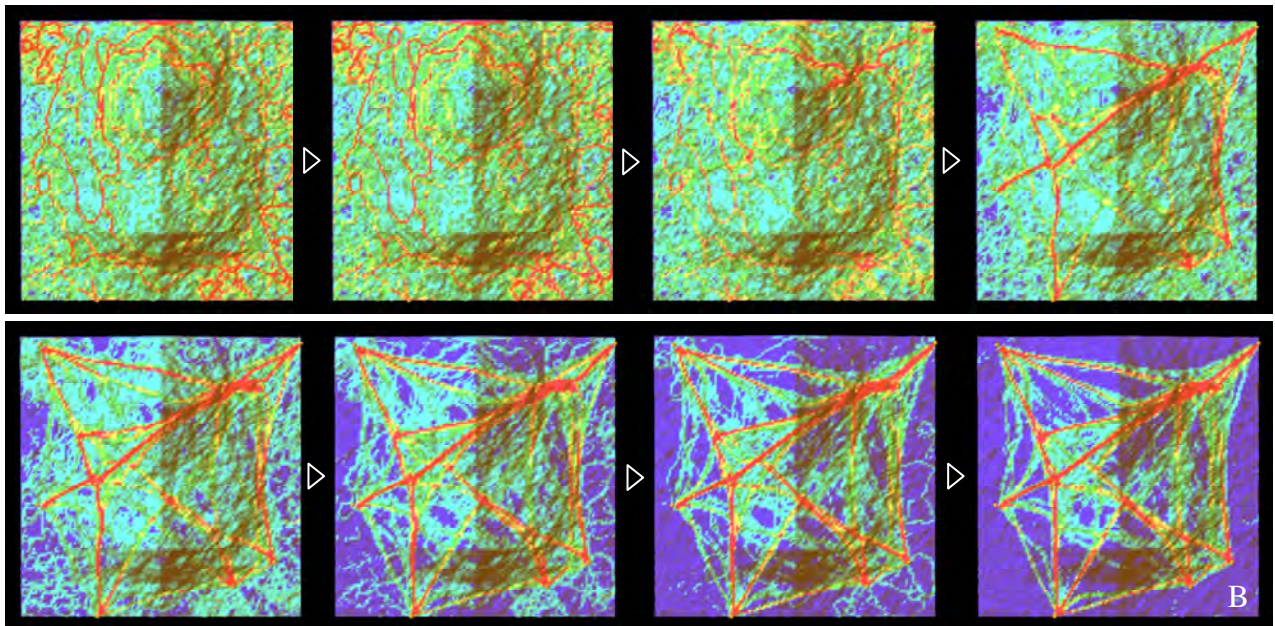
```
//BAD all blue
(T=55%; S=43%; D=2%)
Agent0.T_factor      = 25;
Agent0.Slope_factor  = 20;
Agent0.ChemRate      = 0.012f;
Agent0.TransMoveS    = 1.0f;
Agent0.slopeDistMax  = 1.5;
Agent0.MaxSlope      = 0.8;
Agent0.ChemMaxdist   = 6;
Primitive.decayRate   = 0.9995f;
Primitive.diffusionRate = 0.0f;
```

See Fig.6.23-D

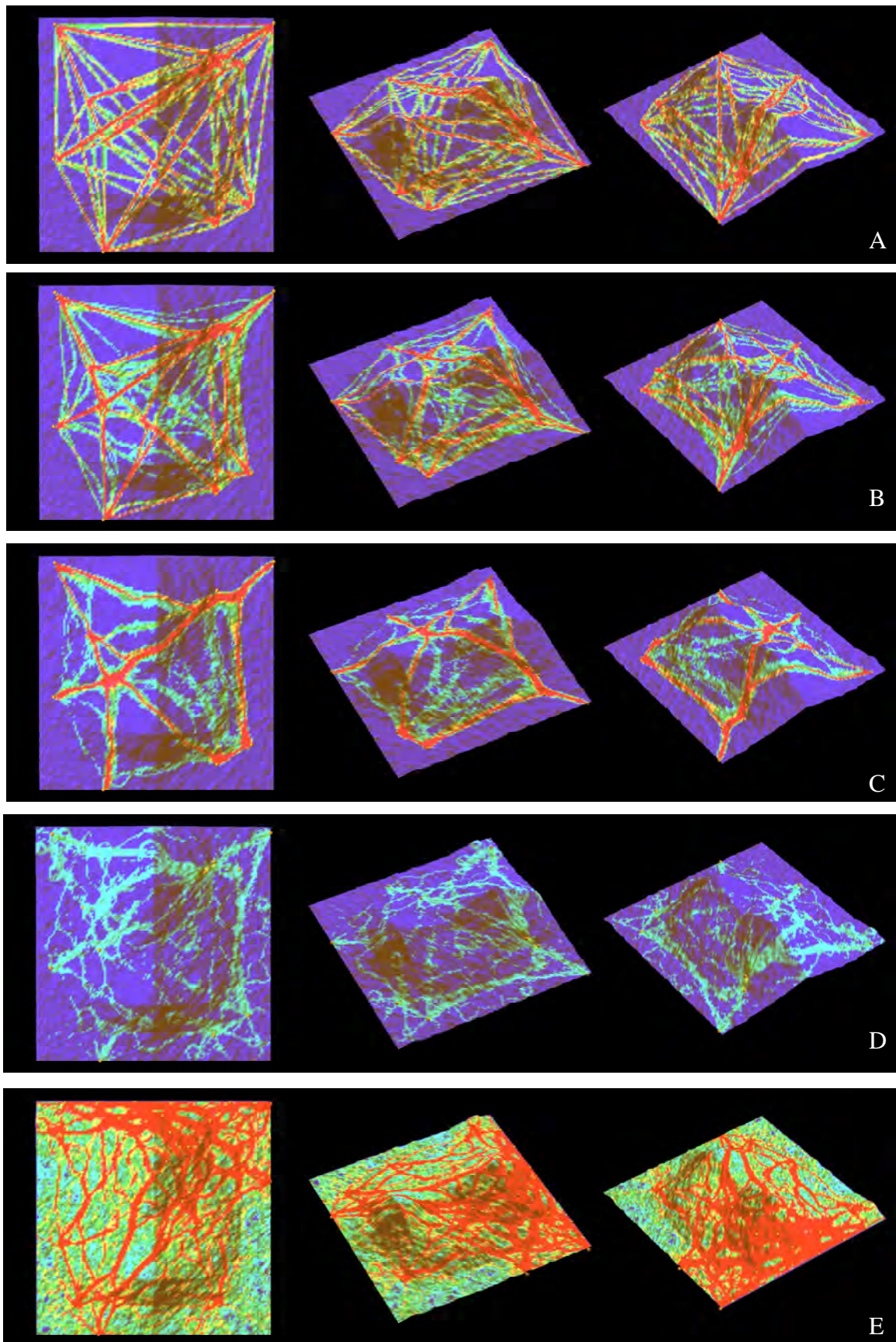
```
//TOO THIN BLUE
(T=27%; S=42%; D=35%)
Agent0.T_factor      = 1.65;
Agent0.Slope_factor  = 0.6;
Agent0.ChemRate      = 0.012f;
Agent0.TransMoveS    = 1.0f;
Agent0.slopeDistMax  = 1.5;
Agent0.MaxSlope      = 0.8;
Agent0.ChemMaxdist   = 6;
Primitive.decayRate   = 0.995f;
Primitive.diffusionRate = 0.002f;
```

```
//BAD all blue
(T=17%; S=83%; D=0.1%)
Agent0.T_factor      = 100;
Agent0.Slope_factor  = 500;
Agent0.ChemRate      = 0.012f;
Agent0.TransMoveS    = 1.0f;
Agent0.slopeDistMax  = 5;
Agent0.MaxSlope      = 0.5;
Agent0.ChemMaxdist   = 6;
Primitive.decayRate   = 0.9995f;
Primitive.diffusionRate = 0.0f;
```

(Note: D is set as a constant value 1.0 above. (Actual D value in percentage = $D/(D+S+T)*100$)



Figures 6.22 – Gradual transformation of trails and city emergence.



Figures 6.23 – City emergence and street networks with various parameter values.

4) Building Behaviors

Buildings are also important components of the program. So far, I have not referred to buildings, choosing to clarify each component concept one by one, but generations of buildings are not separated from the other contexts of generations. In fact, generations of buildings are assumed to be heavily inter-related with the generations of streets and behaviors of agents. They all have to be part of the self-organizing computation.



Figure 6.24 – Developments in Mykonos, Greece, from the 17th to mid-20th century.

Buildings are treated as a separate class of object called “bldg” inside the system. Once conditions for buildings to be erected are met, a bldg-object is instantiated at the location and with the size that is specified. Emergence of buildings is dependent on environmental potentials. Bldg-class objects have their own variable called age, and age-variables record the duration of buildings’ life-time based on a discrete unit time scale of the simulation. Buildings over one hundred years of age have a random rate of chance to disappear.

The environmental potentials are mainly frequencies of traffic and topographical conditions of a terrain. The frequencies of traffic at each patch of the terrain are also considered as an indication of population density of the area. In this system, the measure of traffic intensity is based on how many agents pass that particular location of the patch over time, and this value is reduced at a certain rate of decay if no agents pass the location for a while. Thus this value can represent activity level of the site as well. When this value is higher than a certain threshold value, the site is considered as a potential location for buildings. In this system, when the site activity level exceeds a certain minimal value (which is a dynamic threshold variable interrelated with the environment's overall growth), a building has a chance to be built at the site by a random probability. This random probability is also influenced by the site's activity level. (For example, the probability for building generation exponentially increases as traffic intensity of a patch gets higher.) The sizes of the buildings are defined by the activity level of the site. I used a negative exponential function to define the height of buildings from the chemical value of the patch. The negative exponential function restrains the growth of building height to some upper bound number and seems to represent the variations of buildings' heights found in real-world settlements.

```
//Building generation based on Traffic Intensity
if (chemical-level >= chembldg){
  //Avoid slope Below 30degree
  if (getAngle(Surface.Normals[i][j], Vector3(0,0,1)) < 0.5){
    Bldg b = new Bldg(Surf.Vertices[i][j].x, Surf.Vertices[i][j].y);
    Form1.Buildings.Add(b);
    //for height of Buildings
    b.H = new Vector3(0, 0, logistic(ch) / 2.5);
  }
}
```

The pseudo-algorithm related to building generations.

```

//Building generation from Cities
foreach (CITY city in cities){
    //generate at random location within a unit dist.
    double x = city.pos.x + rand()*Cos(2*PI*rand());
    double y = city.pos.y + rand()*Sin(2*PI*rand());
    //Avoid slope Below 30degree
    if (Vector3.getAngle(Surf.Normals[x][y], Vector3(0,0,1)) < 0.5){
        Bldg b = new Bldg(Surf.Vertices[x][y].x, Surf.Vertices[x][y].y);
        Form1.Buildings.Add(b);
        //for height of Buildings
        b.H = new Vector3(0, 0, logistic(ch) / 2.5);
    }
}

//Agent's Behavior to kill buildings in the way
public void KillBuild(){
    foreach (Bldg b in Form1.Buildings){
        if (Vector3.dist(pos, b.pos) < 0.05){
            Form1.killlist.Add(b);
        }
    }
}

//BUILDING Behaviors
foreach (Bldg b in Buildings) {
    if (time % 10 == 0){
        b.age++;
        b.step(); //Building-Align( )and so on.
        b.draw();
    }
    if (b.age > 100)
        killlist.Add(b);
}
foreach (Bldg b in killlist) {
    Buildings.Remove(b);
}

```

The pseudo-algorithm related to building generations.

Slopes of the site are another criterion for building sites. If the maximum slope of the site is above a certain degree, it is quite likely that no settler is willing to build any structures at such a steep slope. This maximum value can be dependent on regional tectonic cultures, available materials and technologies, and climatic conditions. I set this to 30 degrees based on conventional standards as a maximum slope angle for buildable areas.

5) Negotiation between Agents and Buildings

After the environmental potentials are met, buildings start to emerge. At first, the terrain is completely unoccupied and deserted. The first settlers start finding better areas for their settlements and begin building. The locations of buildings should not directly block the pathways of agents; buildings will negotiate their locations with agents. If buildings are generated by environmental potential to be built right at an agent passage point, then they need to disappear (by literally increasing their ages). If they are close to the agents' path, buildings align themselves to agents' heading directions and shift their locations to avoid direct collisions.

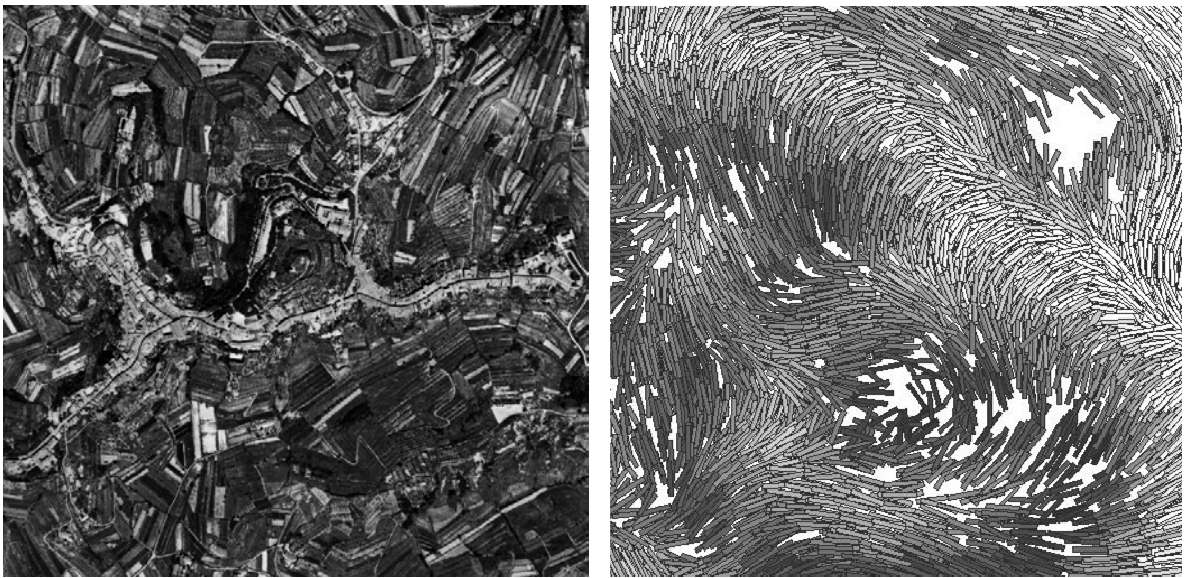


Figure 6.25 – Aerial view of San Miniato (Fanelli, 1990) (left) and self-organizing computation simulation using simple alignment algorithm by the author (right).

There are several other building behaviors, some of them not as visually effective as the basic behaviors in terms of producing realistic arrangements of buildings. One is an alignment behavior. Buildings look around their neighborhood and calculate the average rotation angle of other buildings within a certain distance away. Then they try to align themselves to the average angle. This is a self-organizing behavior often seen among a

flock of birds. Buildings that are located closer to agents' circulations form a street façade by aligning themselves to directions of streets. Buildings behind those buildings forming street fronts start to mimic the alignments of buildings around them. Synchronization among buildings' rotation angles is eventually expected to produce natural arrays of buildings. This feature was successful when numbers of buildings were relatively low. (See figure 6.25)

```
//get average orientation of buildings within Min-dist away and align
void Building-Align( ){
    Vector3 headSum = (0, 0, 0);

    foreach( bldg in Buildings ){
        If(distance(this, bldg)< Min-dist){
            headSum += bldg.head;
        }
    }
    headSum = headSum.normalize();
    this.head = AlignThisBuildingToVector(headSum);
}
```

The canonical building-align-orientation algorithm

Another idea is to use the bubble meshing methods that I introduced in chapter 3. Instead of circles, rectangles of various sizes push and pull each other and self-organize to find better configurations. This method, too, works well with low numbers of buildings; however, for city design applications that deal with thousands of structures, the time for calculation becomes extremely extended. Buildings need to be subdivided into groups within kernels in order to optimize the calculations.

Once the overall environmental potential grows above the aforementioned minima, cities will have emerged. These cities become new sources of occurrence for buildings. At randomly selected locations within a certain proximity of each city center, new buildings are inserted at a certain rate. If the locations are already occupied, the system will find alternative unoccupied locations.

6.2.7 Experiments

In order to evaluate the results from the system, information from an existing site, San Miniato in the region of Tuscany in Italy, was used. San Miniato is located halfway between Pisa and Florence and is on top of the hills dominating the river plains such as the Arno, Era, and Egola valleys. It is an extremely ancient settlement, dating from the prehistoric era. In the eighth century C.E., it was a small village. By the end of the tenth century, it was already highly populated and surrounded by a moat. In the 12th century under Frederick II, the town was fortified with extended walls and other features for defense purposes.

San Miniato displays typical settlement patterns where generations of building structures and street networks are highly influenced by their topographic conditions. San Miniato has an extremely unusual topographic condition, having three distinct hills, and its urban settlements seem to have some correlations with its landform. Since the system being applied in this chapter is designed to find the correlations between landform and manmade structures, this is a good test environment for it. Having an actual site with a certain spatial scale is useful for objectifying various parameters of the system. Comparisons of results from the system and the actual existing site are presented.

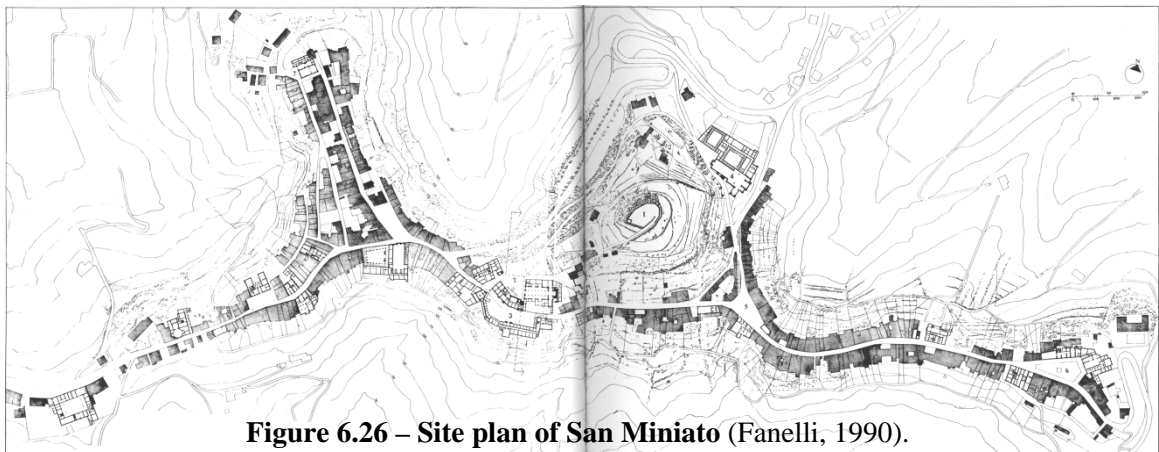
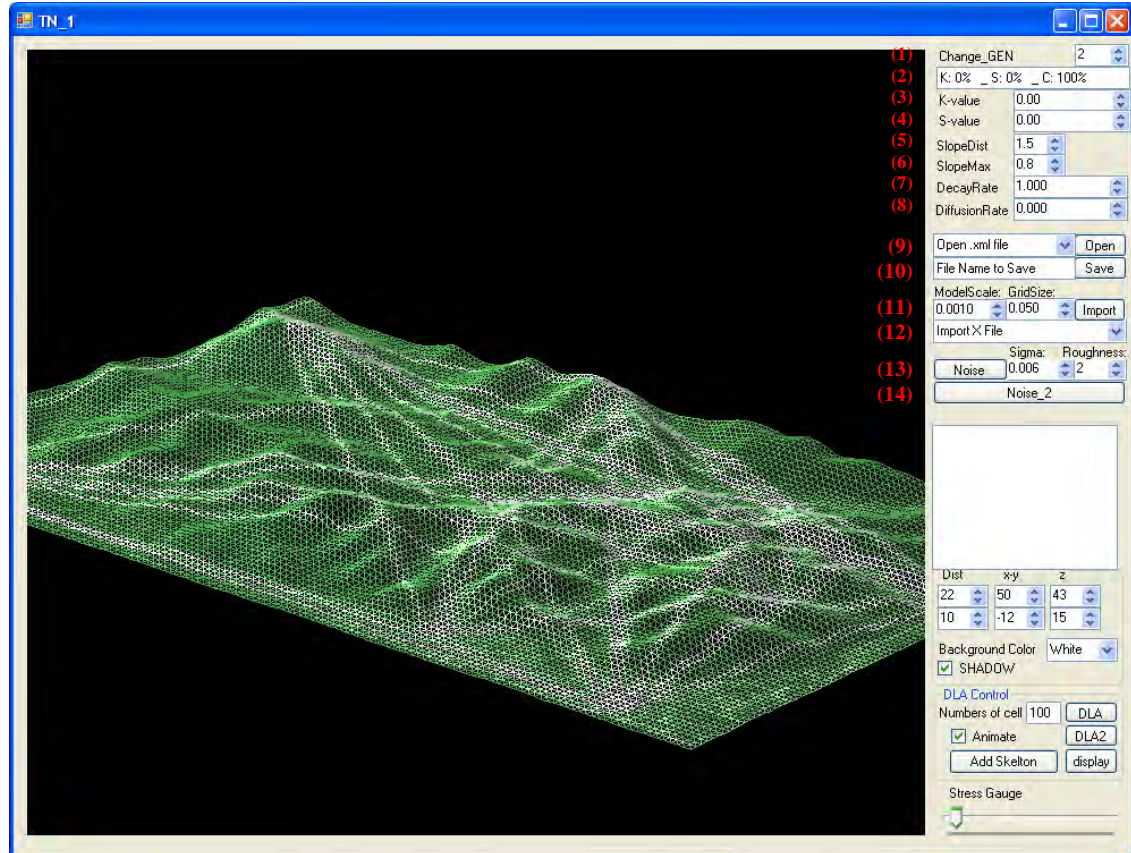


Figure 6.26 – Site plan of San Miniato (Fanelli, 1990).



Figure 6.27 – Views of San Miniato (Fanelli, 1990).

The topographic information of the site was gained from the Google Earth application. Meshes that represent the site were first exported to the SketchUp application, then to Rhinoceros to generate DirectX file format (.x). The DirectX file can be imported into the system's environment by reading vertices and surface normals of the mesh. The original source file does not have extensive detail, so the experiments used meshes with several different levels of fractal noise to see the variations in results.

Graphic User Interface:**Legends:**

- | | |
|---------------------------------|--|
| (1) Generation Indicator | (8) Diffusion Rate |
| (2) T-S-C factors Indicator (%) | (9) Select file name to OPEN |
| (3) T-value (for C=1.0) | (10) File Name to SAVE |
| (4) S-value (for C=1.0) | (11) Model-Scale, Grid-Size for IMPORT |
| (5) Slope-distance value | (12) Select DirectX-file to Import |
| (6) Maximum Slope value | (13) FRACTAL Noise, height & roughness |
| (7) Decay Rate | (14) NOISE function for surfaces |

Keys:

- 'A' := Floor display mode [Checked / Gridlines / None]
 'C' := City display [on/off]
 'I' := Agents display [on/off]
 'J' := Save JPEG file
 'K' := Building display [on/off]
 'Q' := Record movie
 'W' := Display Mode [Shaded / Wireframe / Dotted]

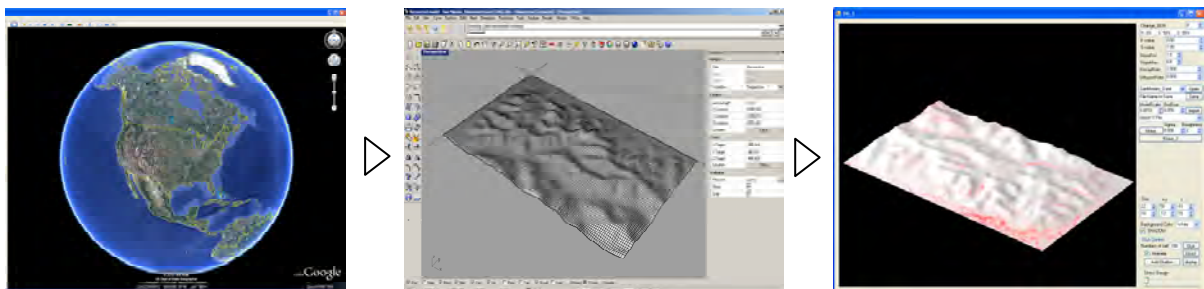


Figure 6.28 – From Google Earth (2010), Rhinoceros (McNeel, 2010), to the author's software.

6.2.8 Results

In the early phases of simulation, results have captured some characteristics that exist in the actual site in the current condition. Urban developments at the northwest side of the map (San Miniato Basso) along the main road (SS67: Via Tosco-Romagnola Est.) are captured from the early stages of the simulation. This is probably due to the flat and open condition of the area that has become a preferable area for agents' activities. Implemented behaviors of agents appeared to be a valid driver with respect to this part of the development. A main artery has emerged along the valley running across the site in the east–west direction. This also exists as a relatively narrower road, Via Calenzano. This is the result of a natural consequence of agents' behavior, since agents prefer to walk on level paths due to their S-factor (attraction to slope). Similarly, I found the minor developments of pathways at the ridges of the San Miniato (hill) area. The system created several built structures along the path on the ridges. Unfortunately, the system's resolution was not fine enough to capture the further detail of this area. Developments in the southeast area of the simulation were not recognizable in the actual one. One possible reason is that the model has a boundary condition which does not exist in the real-life site. Flows of agents from main arteries return from the boundary of the site, and overflows of these may cause some irregular movements of agents. This issue needs to be resolved for the future explorations.

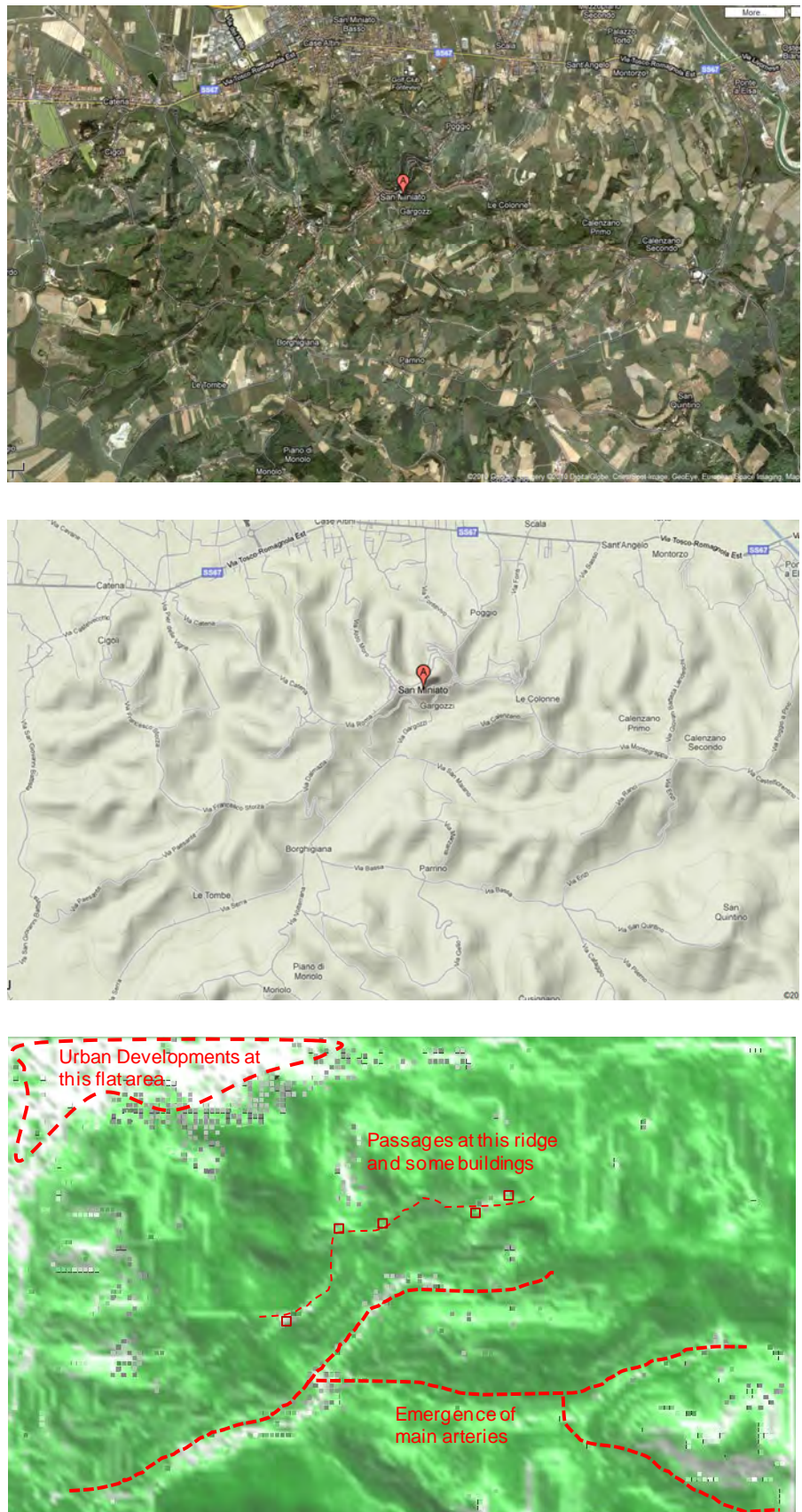


Figure 6.29 – San Miniato from Google Map (top) and a simulation result.

Early Periods

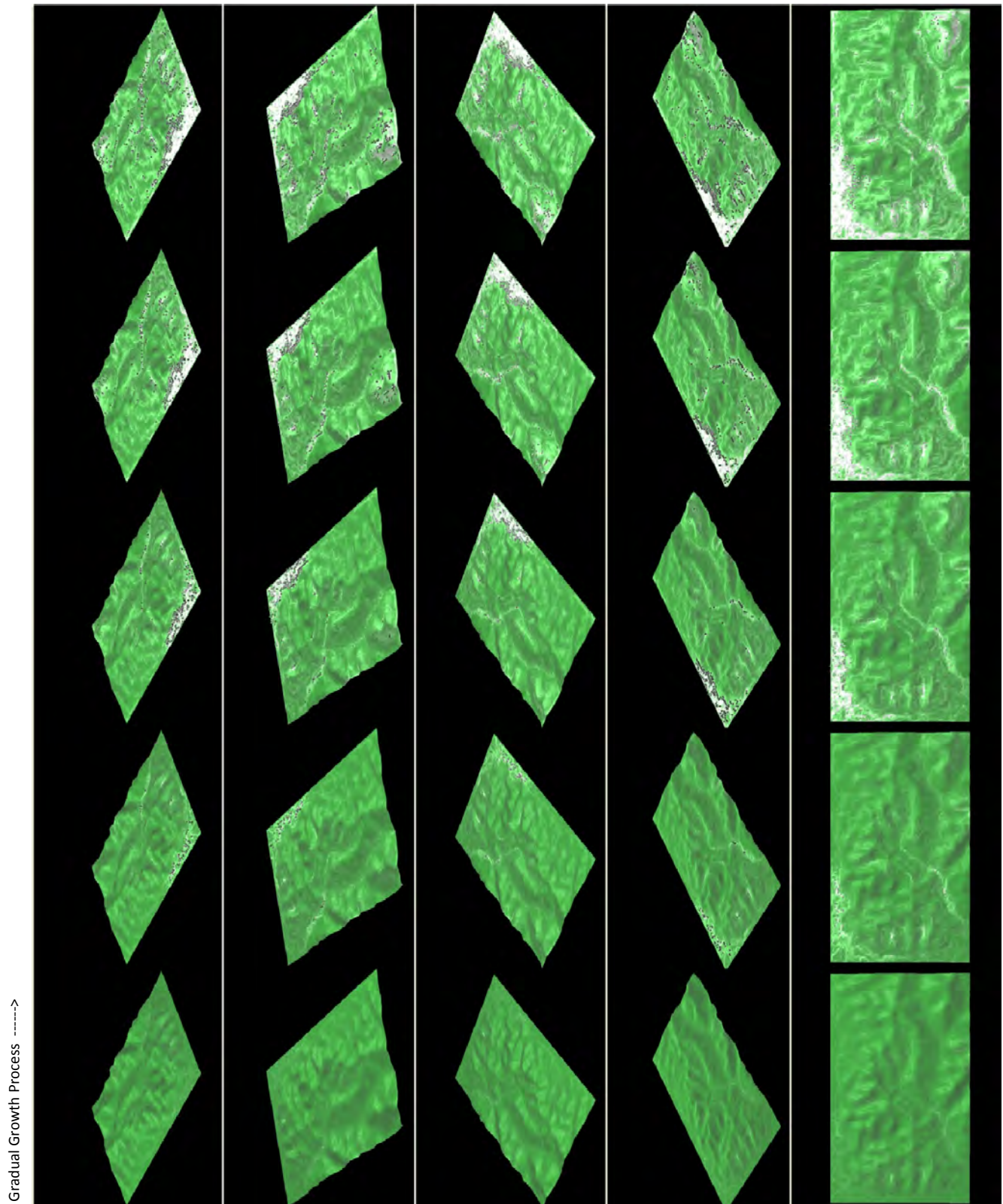


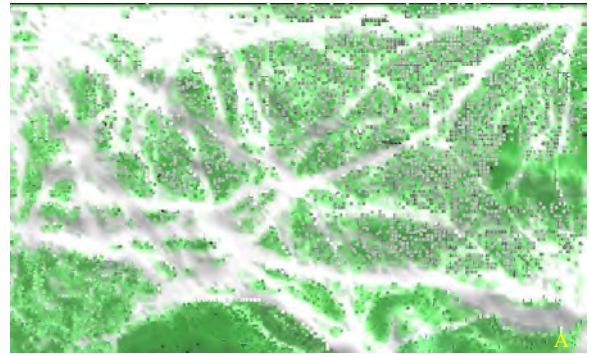
Figure 6.30 – Emergence of Trails (Early Periods)

A) Trail

```

Agent0.T_factor      = 0;
Agent0.S_factor      = 1;
Agent0.MaxSlope      = 0.8;
Agent0.slopeDistMax  = 1.5;
Agent0.ChemRate      = 0.01f;
Agent0.ChemMaxdist   = 6;
Agent0.TransMoveS    = 0.15f;
Primitive.decayRate  = 1f;
Primitive.diffusionRate = 0;

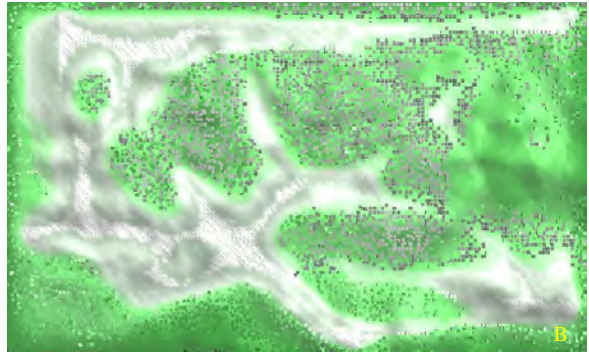
```

**B)**

```

Agent0.T_factor      = 1.65;
Agent0.S_factor      = 0.6;
Agent0.MaxSlope      = 0.8;
Agent0.slopeDistMax  = 1.5;
Agent0.ChemRate      = 0.012f;
Agent0.ChemMaxdist   = 6;
Agent0.TransMoveS    = 1.0f;
Primitive.decayRate  = 0.9935f;
Primitive.diffusionRate = 0.013f;

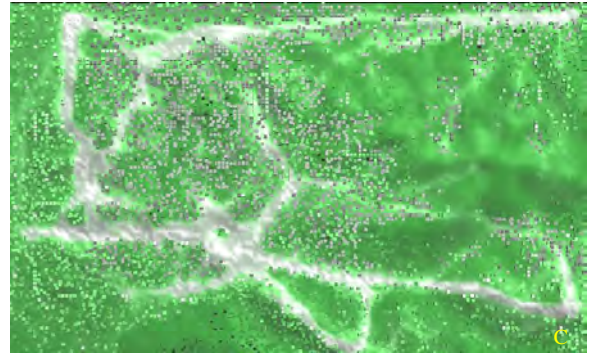
```

**C)**

```

Agent0.T_factor      = 1.65;
Agent0.slope_factor  = 1;
Agent0.MaxSlope      = 0.8;
Agent0.slopeDistMax  = 1.5;
Agent0.ChemRate      = 0.012f;
Agent0.ChemMaxdist   = 6;
Agent0.TransMoveS    = 1.0f;
Primitive.decayRate  = 0.9995f;
Primitive.diffusionRate = 0.0f;

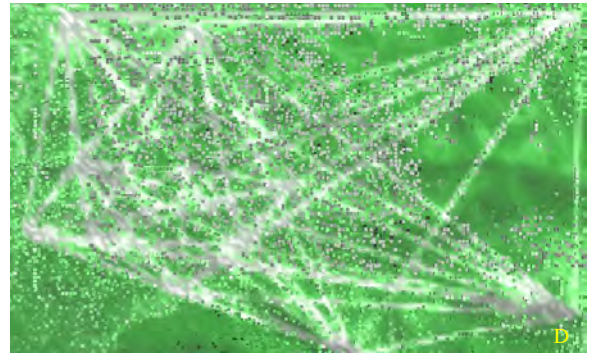
```

**D) Direct Path**

```

Agent0.T_factor      = 0;
Agent0.slope_factor  = 0;
Agent0.MaxSlope      = 0.8;
Agent0.slopeDistMax  = 1.5;
Agent0.ChemRate      = 0.012f;
Agent0.ChemMaxdist   = 6;
Agent0.TransMoveS    = 1.0f;
Primitive.decayRate  = 0.9995f;
Primitive.diffusionRate = 0;

```

**E) Minimal Path**

```

Agent0.T_factor      = 0.85;
Agent0.slope_factor  = 0;
Agent0.MaxSlope      = 0.8;
Agent0.slopeDistMax  = 1.5;
Agent0.ChemRate      = 0.012f;
Agent0.ChemMaxdist   = 6;
Agent0.TransMoveS    = 1.0f;
Primitive.decayRate  = 0.9995f;
Primitive.diffusionRate = 0;

```

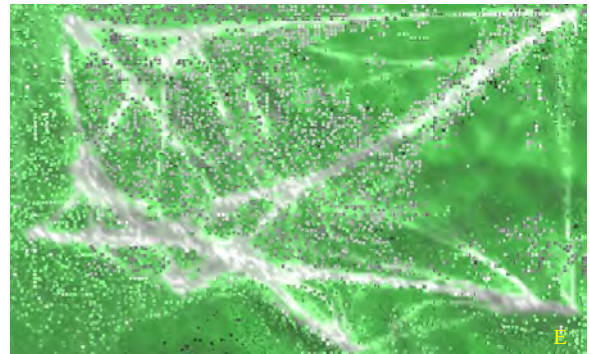


Figure 6.31a – City Growth Phase with 5 different parameter settings.
D is set as a constant value 1.0 above. (Actual D value in percentage = $D/(D+S+T)*100$)

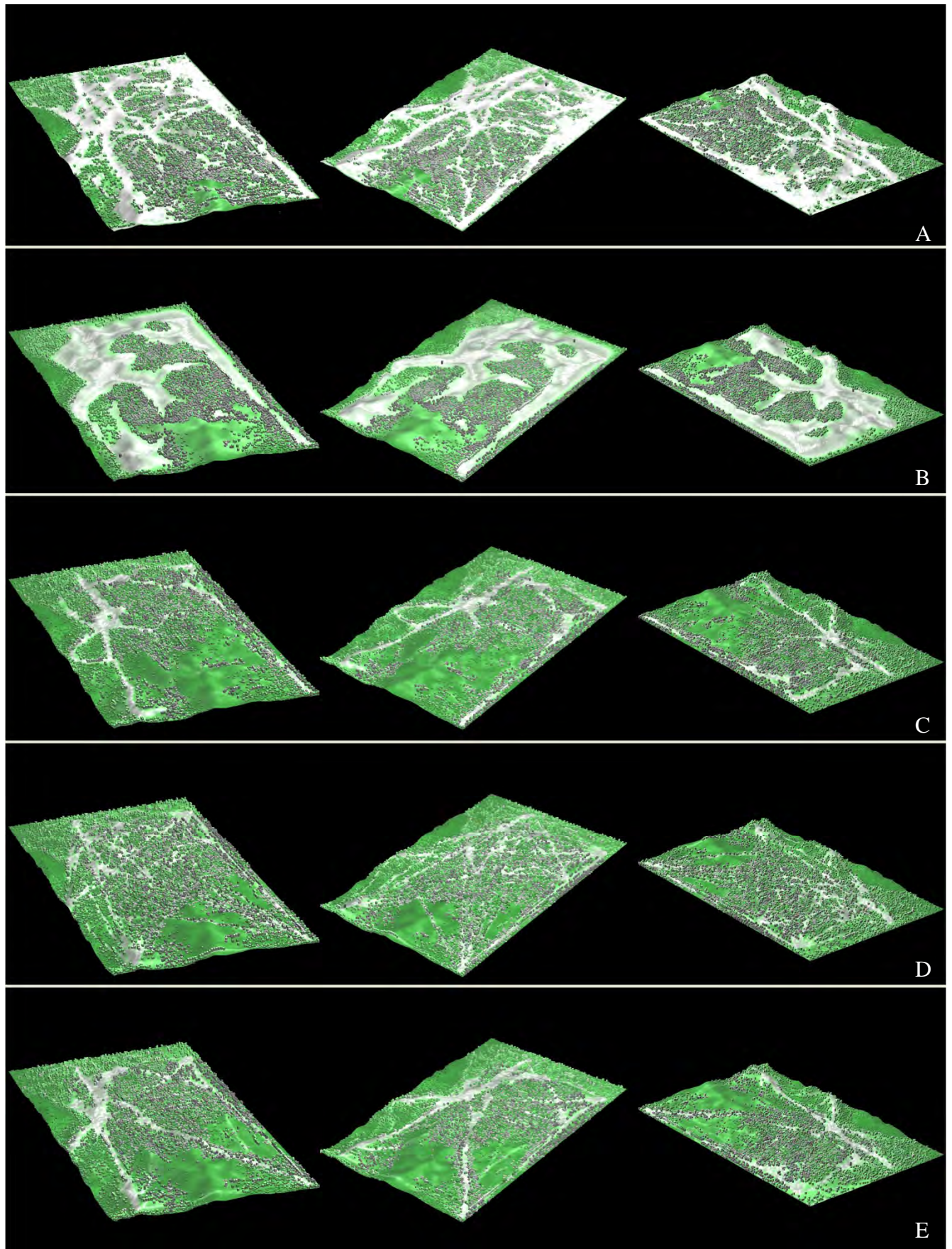


Figure 6.31b – City Growth Phase with 5 different parameter settings.

City Growth Phase

The second phase of the experiments concerns the growth after the extraction of the candidate locations for cities (or population-concentrated areas). This second phase of the experiment is in a speculative domain. In this phase, I assumed that the traffic flows of agents are constantly accumulated and that cities are subject to a continuous demand for growth. (People continuously come into and out of the site area, and the decay-rate and chemical-rate of the site are set to maintain the site's active state.) Five schemes shown here all display an extreme urbanistic future for San Miniato.

If I did not provide any attractions to traffic intensity of terrains, the system continued to grow trails (A). The branching of trails looks natural and this scheme reflects the real scale of the terrain fairly well. Emerging hierarchies among thicknesses of trails are also differentiating major arteries and secondary passages. The buildings eventually filled major islands on the map densely. This pattern looks similar to many cities in Europe or Asia where infrastructures have been gradually developed over long periods of time without experiencing any catastrophic interventions.

Paths that were produced only from an attraction to destinations formed direct paths (D). Although this result does not reflect the scale of the terrain very well, this type of layout can be seen in railway tracks or highway layouts at larger scales.

The other schemes (B, C, and E) are in-between the two preceding schemes in terms of the parameter settings. I obtained bundling arteries at some locations. These results are more economical in terms of the construction of roads since they have a shorter overall length than the trail scheme (A). Depending on the slight differences in parameters, some sub-arteries' configurations vary.

Figure 6.32a – City Growth Phase: Scheme A (Trails).

Gradual Growth Process of Trails and Buildings ----->

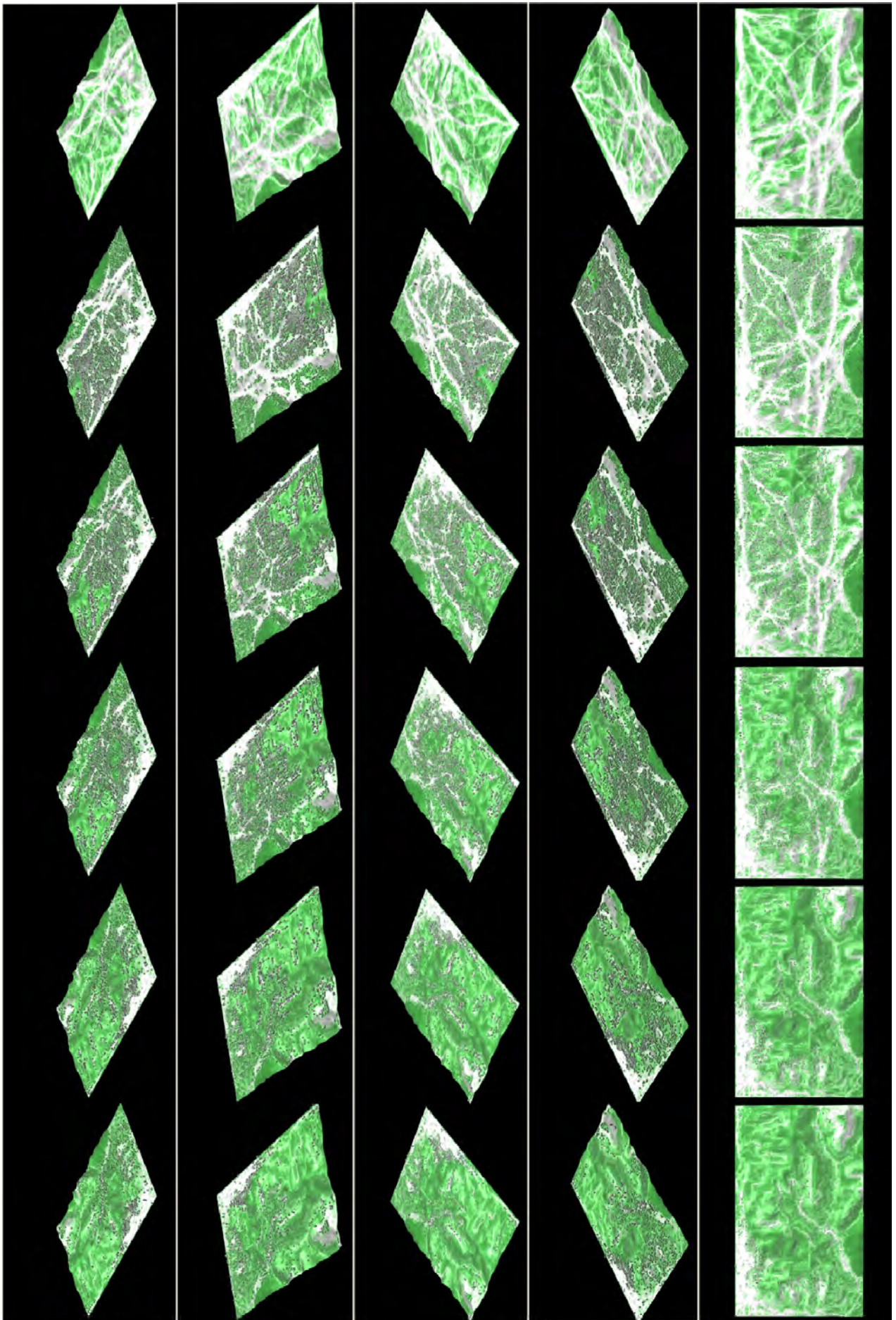


Figure 6.32b – City Growth Phase: Scheme B.

Gradual Growth Process of Trails and Buildings ----->

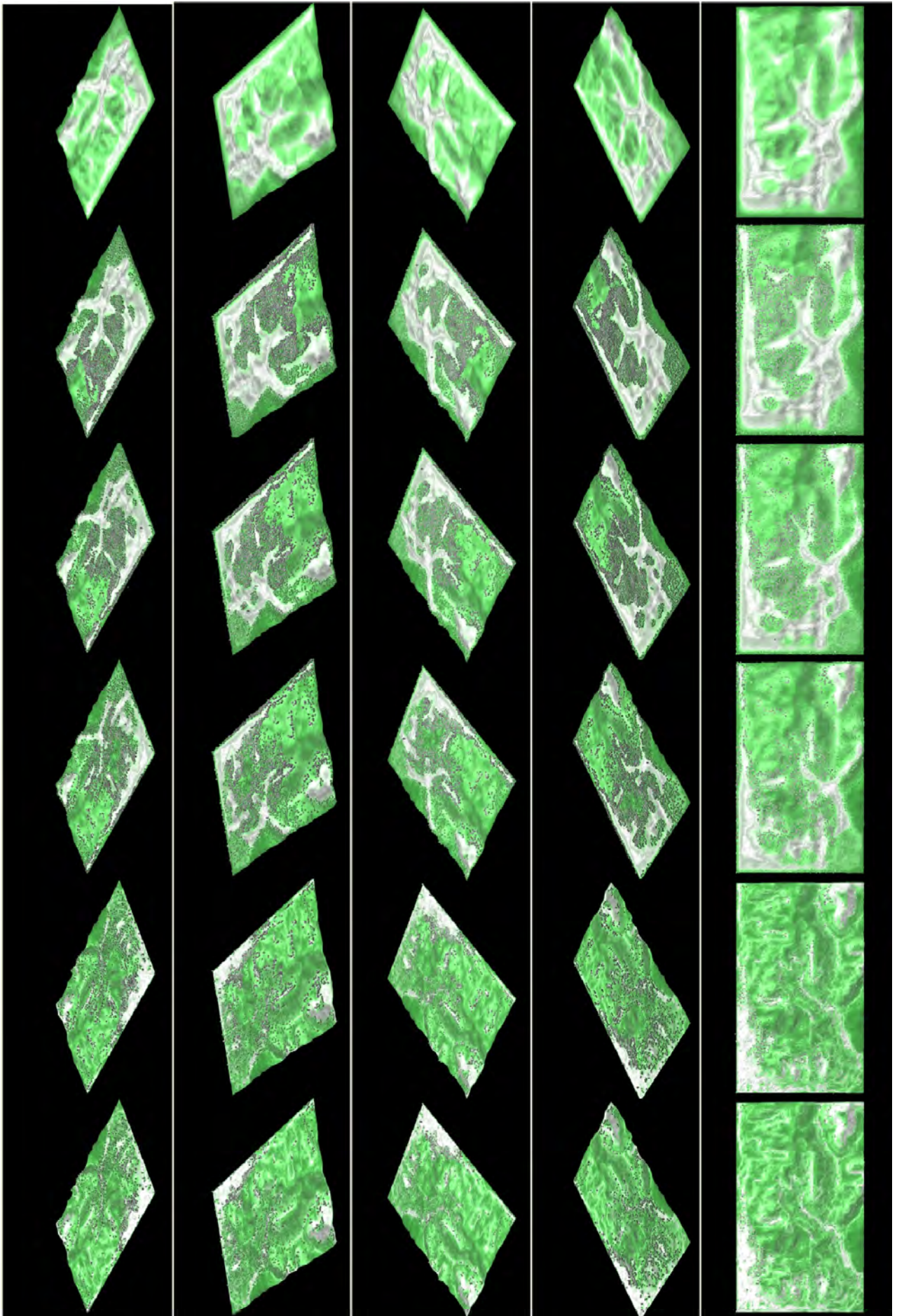


Figure 6.32c – City Growth Phase: Scheme C.

Gradual Growth Process of Trails and Buildings ----->

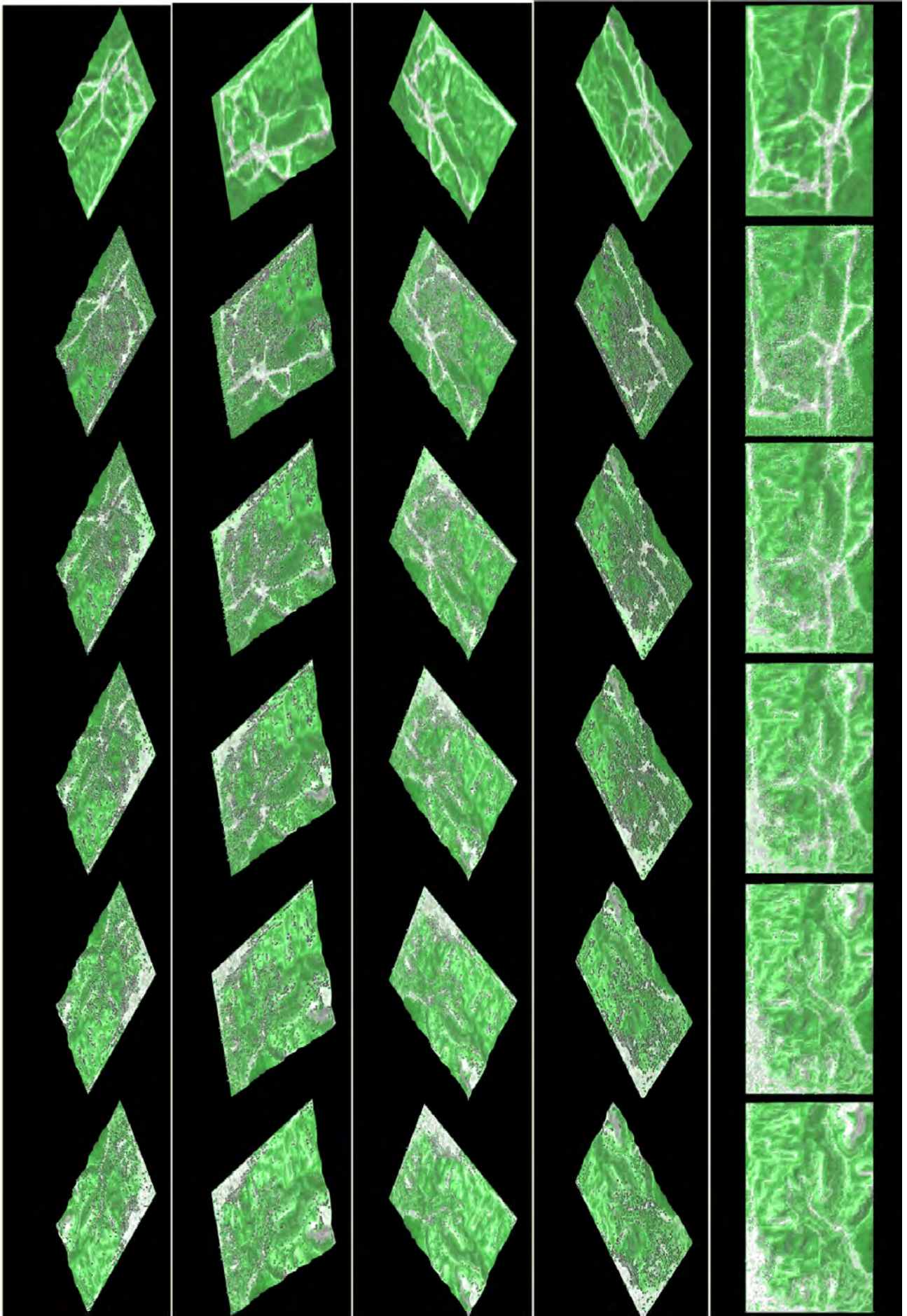


Figure 6.32d – City Growth Phase: Scheme D (Direct Path).

Gradual Growth Process of Trails and Buildings ----->

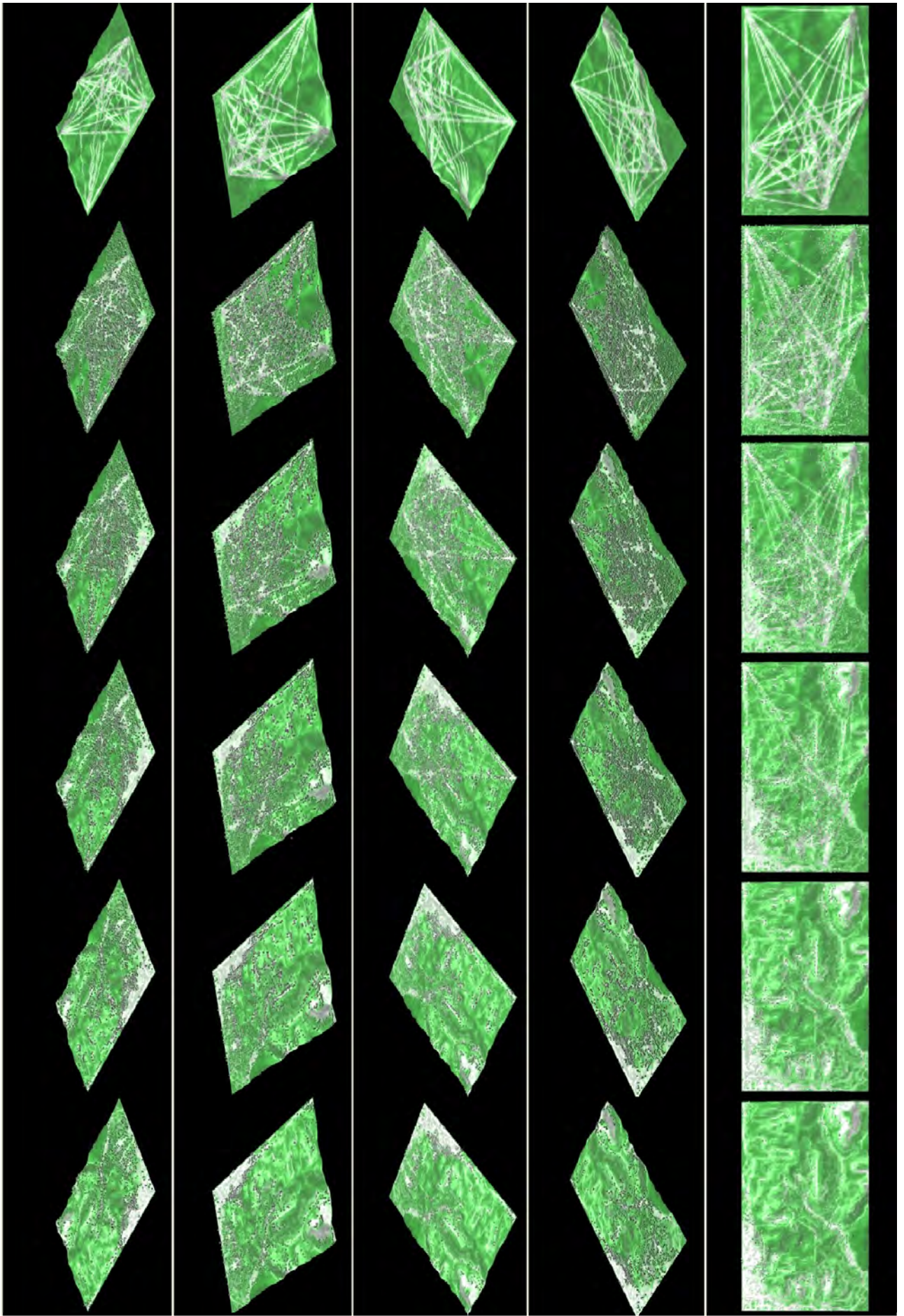
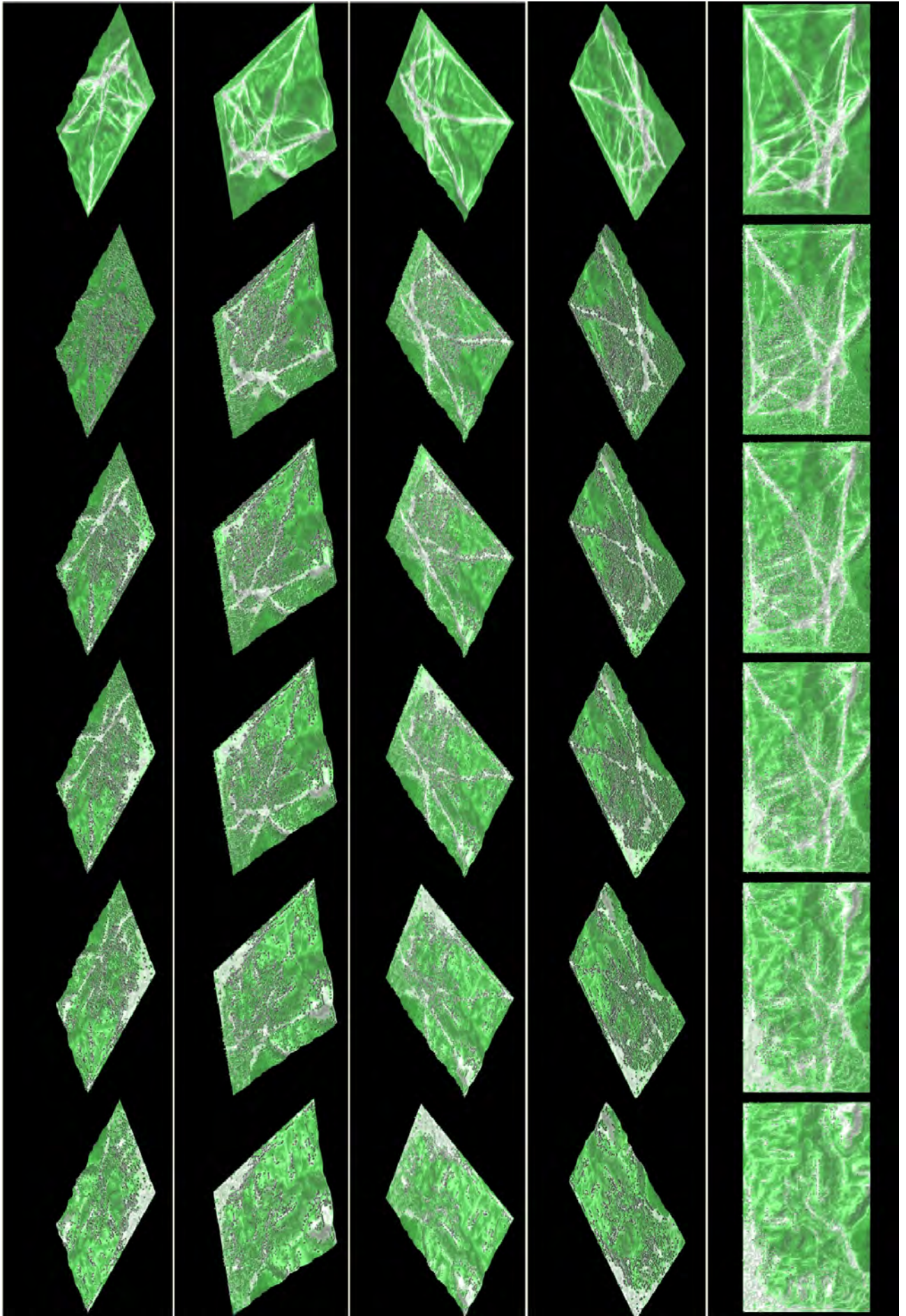


Figure 6.32e – City Growth Phase: Scheme E (Minimum Path).

Gradual Growth Process of Trails and Buildings ----->



6.3 Conclusion

Agents in this program represent possible pedestrian movements in an abstract manner. They can be literally interpreted as pedestrians walking on a terrain at the time of the simulation, but they can also be interpreted as abstract entities that record intensities of activity level based on locations. The original interest was to use them as an empirical tool that could extract behavioral tendencies. Representation of their movements was not the primary purpose of the system. But by taking advantage of parallel computing, multiple agents' reactions are actively used to produce new designs.

Agents are influenced by their environment. However, they can also influence their environment in turn. Environmental potentials are dynamic values that are constantly updated by agents' behaviors and become incentives to change their environment. Moreover, these changes in the environment can stimulate agents to shift their future movements and behaviors. Changes caused by some agents can possibly influence others in their environment as well. This chain propagation of events enables the system to have an open feedback loop between agents and environments. This non-linear feedback is the key characteristic of the system that makes it a generative computational system. This characteristic may lead to producing a self-organizing global spatial structure from local interactions among agents and the environment.

6.3.1 Degree of Reliance on External Knowledge

Many factors in this project are generatively derived from the system itself using self-organizing computation techniques. Unlike other shape-based applications, such as L-

system and shape grammars, the system uses its own components' indirect interactions, and expects the self-organization of large-scale spatial structures. Instead of imposing specific geometry or topological configurations, this system's design variables are reduced to simple key design factors, such as threshold parameters, that induce certain actions among agents. These parameters are among the few factors that the system still requires as inputs.

Due to this high dependency on parameter settings, emergence of large-scale structures is not at all guaranteed in this system. Slight changes in the proportions of parameters for T, S, and D-factors lead to very different results. Even worse, for some cases, if incremental rate and decay rate of chemicals are not balanced well, the results are either explosions of chemical values all over the site or extinction of all activities. Aforementioned geometry-based systems are able to produce more detailed representations because they possess repertoires that are designed by humans from the outset. This lack of detail in our simulated representations also results from the limits of currently available speeds of computation. Greater levels of detail and sophistication will require intensive graphic memories and a more efficient structure for algorithms. Higher processing speed and power of computers definitely promises improvements for self-organizing computation.

6.3.2 Scale of Space and Time

Scale of space and time is one of the most critical issues relating to this particular application. The system sets model-scale and unit-grid-size when importing topographical information from actual site data. These two parameters consequently set

the scale relative to agents and the environment. Agents' movements and behaviors are designed to reflect a certain scale of actions. Agents' cone of vision is one such critical parameter that changes the results based on unit-grid-distance. There must be an optimized scale factor between a model and a real-life environment. After a great number of trials, after witnessing many results that are similar to real-life scenarios, I came to the conclusion that finding the right scale factor that naturally synchronizes two worlds is the most critical part of these self-organizing design experiments.

At this stage of the software development, I could only empirically verify these points of synchronization by testing and adjusting various parameters, and this world inside the system could never be a complete representation of the actual world. Seeing the world through this over-simplified model framework may become a target of criticism. However, I believe that finding a few (but the most critical) factors that lead to the self-organizing phenomena of interest is the first step toward unveiling the principles behind these phenomena. By studying the primary drivers for certain outcomes, there is a chance to understand and decode the principle behind self-organizing pattern formation in many systems. Whether those pattern formations involving human decisions can be reduced to a certain principle or algorithmically representable framework has yet to become clear. I will return to this topic in the next chapter.

6.3.3 Limitations

During the course of socio-economic development, agents gradually grow from trail-seeking settlers to merchants touring through various city centers dispersed in the

environment. Agents change their behaviors from the original wandering movements based on momentary reactions to more mature states using itineraries. The results of street patterns also reflect these changes. Original trails are more organic and almost look like animal trails, whereas ones that developed from agents' more mature states are more goal-oriented and ordered.

Agents' means of transportation are one thing that has not been adequately addressed previously. At the time of the emergence of cities, they have more direct purposes and paths to their destinations, and they may start to rely on faster and more robust means of transportation. In the experiments, it was attempted to address these changes by, for example, reducing the factor for slope attraction and increasing attraction to their destinations, which eventually leads to production of a more directed network similar to highways and railways. The representations of agents are kept as abstract entities and being too specific about these details is avoided.

However, transitions to automobile or railway systems will have major influences on later developments of the environment. For example, railway stations might bring prosperity to the nearby regions; however, railway lines often physically disconnect and separate some areas in undesirable manners. In real-life scenarios, these changes by human interventions are discrete and discontinuous transitions rather than gradual ones implemented in the system, and due to that discrete and discontinuous nature of the transitions, some environmental changes can occur abruptly.

Maps in Figure 31 (Ferguson et al., 1990; Brant, 1994) show continual transformation of structures in a Zuni pueblo village. These structures gradually dispersed as the needs for defense declined and vehicular transportation found streets useful. These are changes caused by specific physical and technological developments, and simulation of these self-organizing pattern formations requires more specific implementation of major components. At this stage, the abstract and conceptual nature of the system has limitations with respect to capturing some details.

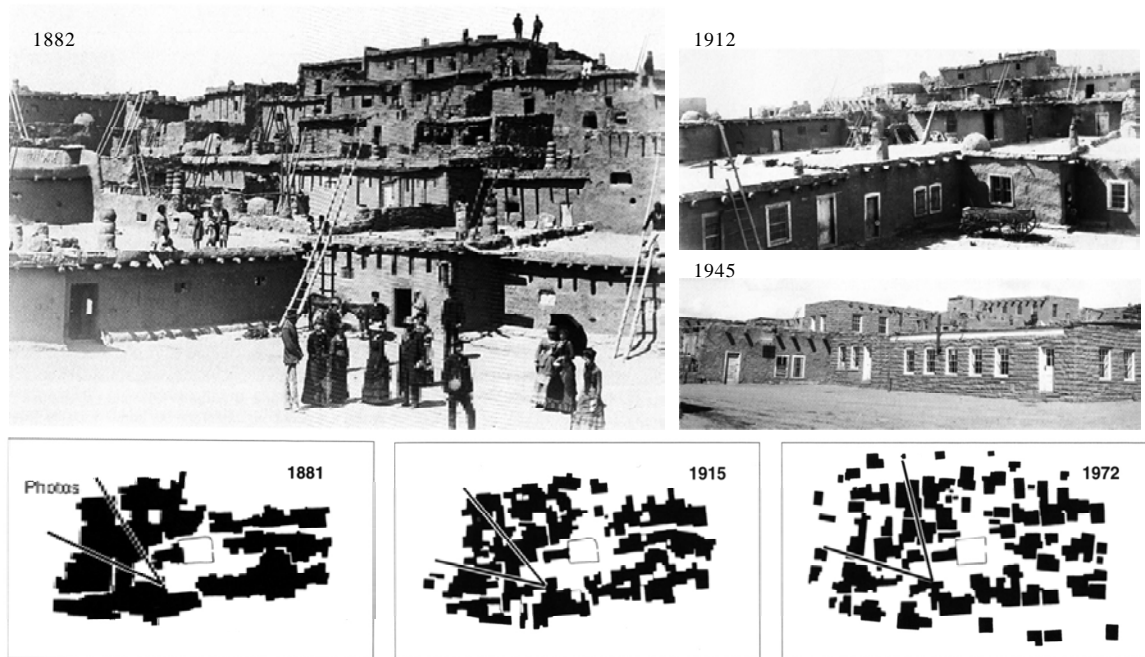


Figure 6.33 – Continual transformation of structures in Zuni pueblo village. Maps created by Mendelev, V., Kroeber, A., and Borchers, P. from (Ferguson et al., 1990; Brant, 1994).

Another critical issue is the difficulty of knowing the legitimacy of the results from the system. It is difficult to know for sure whether the system's state is already mature enough to introduce new behaviors for agents. I have tried the simulations with many different seed values for random number generators. I expected to see the same, or

relatively close, results for the locations of emergent cities; however, I found some perturbations based on the seed values. Although agents start with different seed values, if I use sufficient numbers of agents and trials, the results should display some cohesion between behaviors and environments. I allow some levels of perturbations in results due to the nature of stochastic simulations. But some inconsistencies found in results may imply that the numbers of trials by agents might be insufficient to shift their behaviors. The results might be overly biased by initial stages of the simulations, which are likely to be influenced by the different seed values. If I continue the simulations long enough, this bias should be weakened, and results that are essential to the correlation among agents and environments should prevail over the initial biases.

6.3.4 Future Work

Emergence of hierarchy among built structures is one implementation that I would like to add to my system. There are several promising computational strategies for implementing the emergence of hierarchies. One is full implementation of the bubble mesh method introduced in chapter 3 for buildings and roads. Buildings push and pull each other, based on given territorial parameters and adjacent road configurations, to self-organize themselves. This implementation of active form-finding behavior is a technically feasible level of implementation. I already have alignment algorithms for all buildings and roads; however, this technique was computationally too time-consuming for numbers of buildings exceeding 5,000. Tests on smaller-scale areas can be conducted.

The buildings are represented as rectangular boxes with various heights based on the population density of the area; however, their growth does not always need to follow continuous functions of growth. Clusters of smaller buildings can be turned into mid-rise buildings as the economy and needs of the area increase. Densely packed bundles of mid-rise buildings can be turned into a single high-rise building at some point in the growth due to the economy and spatial needs of the sites. Such events triggered by human interventions are discontinuous events and do not always follow continuous growth patterns. These interventions can also be influenced by types of occupancies and the roles of buildings in regions as well. Emergence of new buildings' physical orientations and types of programs can further affect the later development of regions. Some buildings can turn into monumental landmarks, and the visual impact of those can be a major stimulus for agents to execute new actions. As with biological growth starting from a single cell and later turning into specialized body components, morphogenic interpretation of city development can be a challenge. Simulation starts by aggregating homogeneous components as built structures, but these aggregations turn into more specific programs and purposes, such as residential or commercial, as the simulation develops. Simulation of building developments in modern society and more contemporary phases of agents' behaviors merits further investigation. A modern phase of this application is anticipated.

Evolutionary game theory

I would also like to note that evolutionary game theory is another potential computational technique for describing dynamics among various building types and their negotiations. Evolutionary game theory considers “repeatedly played games” and introduces the

notions of mutation and selection, whereas traditional game theory was mostly concerned with one-shot or discrete numbers of games. As I noted in chapter 3, Novak and May (1992; 2006) introduced the spatial extension of evolutionary dynamics using cellular automata. Negotiation among different building types can possibly be implemented by using these notions from evolutionary game theory. Survivals and extinctions of various building types in densely populated urban settings can be represented by strategic aspects of evolution based on mutation and selection. Using interactions among different characteristics of architectural program types as payoffs, and simulating their long-term configurative trends, could be one promising future implementation.

Active Learning by Agents

Agents' active learning capability is another challenge for future experiments. I have introduced changes in agents' behaviors induced by environmental changes. Agents are programmed to produce new behaviors from compounds of a few sets of parameterized primitives. However, agents' behaviors are constrained by the system's limitations and they do not create entirely new behaviors from scratch in this program. To some extent, this would be computationally possible by applying learning algorithms such as neural networks to the self-organizing computation frameworks. Ohuchi (2002) introduced a method to combine a neural network and a genetic algorithm to evolve agents' behaviors. Genetic programming is another alternative that can allow systems to produce new ideas. The author has previously experimented in a different project with using a combination of a GA and a neural network. This method requires extensive computation time and was only feasible for very simple implementations with hardware available at the time.

Chapter 7

Conclusions

Introduction

This thesis has documented efforts to describe computational strategies in architecture and urban design inspired by the concept of self-organization which can be seen from some natural and artificial systems' behaviors. My research was based on the assumption that with emerging complexity of architectural programs, quantities of information, and advanced technologies, there will be more demands for buildings to be able to change over time to adapt to newly emerging needs for different qualities and quantities of architectural and urban-scale components.

The proposed methods are still in abstract forms and are not ready to produce highly detailed instances of architecture. However, they show some advantages of self-organizing computation through the conceptual frameworks of the projects. Self-organizing computation's ability to describe systems that need to change over time

(growth), its ability to discover new goals and objectives, and its suitability for searching large design spaces are main characteristics that can be recognized from experiments in this thesis.

In the case of closed systems with limited reconfigurability, decentralized control systems operated by self-organizing logics may provide robustness and flexibility over conventional centralized control systems. Operable shading devices can be a simple example that falls into this category. However, the true advantages of self-organizing computation are in the applications of its logic to extensible reconfigurable systems that are not limited to finite predefined configurations – systems that can grow over time and acquire new objectives based on interactions with their ever-changing environments. A “city” is a good example of this category despite its slow pace of growth sequence. A city has a few basic components – buildings, streets, transportation infrastructures, and so on – and its spontaneous developments are the results of interactions among these fundamental components and their environments.

There are emerging cases of extensible reconfigurable systems on a smaller physical scale, closer to that of buildings and with the shorter span of time that buildings require for their developments. This is due to the advancing technologies to build larger structures faster and to make them more active and responsive by the use of actuation and sensing devices. These emerging new building types are still embryonic, and we have yet to see them in actuality; however, realization and design strategies of such systems may benefit by an active incorporation of self-organizing computation as introduced in this thesis. The proposed methods’ applicability is still in the speculative domain of future

architectural developments, yet this thesis has tried to analyze the possibilities of self-organizing computation through clear conceptual experiments.

7.1 Two Difficulties in Computational Approaches to Generative Design in Architecture

In the last three chapters, I have introduced and reviewed various self-organizing computational techniques applied to architectural design contexts. Most of them remain in highly conceptual stages, yet they can, in principle, be extended to more specific contexts of architectural problems. In general, problems in computational approaches to generation of architectural designs involve two issues. Firstly, the search spaces for solutions are too large if we include all possible design configurations. For instance, if we simply think about shapes of a table, even with a single material, there are too many formal possibilities. This is even more true when considering a range of materials in combination. Secondly, many objectives for architectural designs are multi-dimensional, and often objectives conflict. For example, designing a large tabletop surface with skinny and less visible support structures are two conflicting objectives. Multiple objectives at odds with each other often lead searches into non-linear problems.

7.2 Two Common Approaches

Moreover, in addition to these two issues, if we start to consider adaptation to dynamic changes of environments and active growth for objects being designed, finding the right computational framework becomes a complex task, and the time that it takes to compute increases exponentially. In order to address these difficulties in a computationally

feasible manner, we need to strategically reframe the problem. There are two possible approaches for this task.

(1) Reduce Possibilities and Choose among a Smaller Subset

One approach is to reduce the problem's framework to a simpler form in order to reduce the size of the search space. By simplifying the problem's framework, we can expect greater reduction in the size of the search space. A smart selection and clever implementation of objectives can be one way to simplify the framework. Complex architectural problems dealing with numbers of formal variations can, in principle, be reduced to simpler problem formats. For example, architectural plan configurations can be reduced to topological graphs indicating connectivity of each space. We thereby reduce the number of slight perturbations in plan geometries, and the framework of the problem is simplified. Consequently, the search space is greatly reduced and it might be possible to find the right configurations from limited numbers of permutations of schemes based on a process of elimination. In the case of quantitatively representable objectives, such as structural issues, this approach could be effective. Even for some of the qualitative issues, user interactive frameworks that permit users input, such as the aforementioned interactive genetic algorithms, can solve the problem to some extent.

One possible drawback of this approach lies in oversimplification of architectural problems. This method leads to some awkward selections due to the method's inability to capture subtle differences in architectural representations. Oversimplified architectural representation also may not provide correct one-to-one mapping to physical building designs. The system doing the simplification is not always able to distinguish all

categories of architectural programs or components, and sometimes it forces some components to be classified into categories that do not conform to the original characteristics.

The system may be able to provide reasonable results for purely logic-driven problem parameters; however, some problems' search space can still be outrageously too large to calculate even with merely logic-driven parameters. In addition, if all the architectural problems are reducible to some deterministically solvable simple formats, another controversial question emerges: What are the roles of humans in the processes of design? Do we just need to provide concise models, formulate the problems, and wait for the answers from machines?

(2) Employ Methods That Feature Self-directed Solution-seeking Behaviors

Another approach to solving the problem of enormous size of search space is to find clever and efficient search methods. Self-organizing computation and search methods based on stochastic selections such as a selection sequence seen in a genetic algorithm (GA) probably fall into this category. This approach is to accept a problem's framework as is, and to find search methods that are able to actively maneuver inside a fitness landscape to search for solutions. These methods can direct themselves to find better solutions within efficient computational time. The aforementioned circle packing problem using a bubble meshing method was one approach representing this methodology. The search space for packing multiple differently-sized circles within a circle is extremely large. Formulating analytical methods to directly derive the best solution is not feasible with the current speed of any available computers. Simulation

using a bubble meshing method finds better solutions in a reasonable range of computational time by using the physical dynamics of circles pushing and pulling each other. The Nelder-Mead optimization method introduced in chapter 4 is another example displaying this heuristic approach. In the particle swarm optimization (Kennedy and Eberhart, 1995), multiple particles inside the multi-dimensional solution space can collectively and interactively move to find optimum solutions from a fitness landscape.

These systems do not guarantee the best solution from the outset, but provide reasonable solutions within a computationally feasible time. Self-organizing computation accepts problems' frameworks and lets the system search its emergent structures for solutions. These systems have features that can be characterized as autonomous solution-seeking behaviors. Many of these systems, such as GAs, do not need instructions about how to find solutions. Fluctuations within the system are also one attribute of such systems that can sow seeds for the discovery of new solutions inside the vast search spaces. A polytope used in the Nelder-Mead method in Chapter 5 represents this characteristic almost literally. The polytope moves toward a local optimum of a problem by replacing one of its vertices with one closer toward the optimum. In the case of search in 2-dimensional space, the polytope is a triangle climbing toward peak areas of the fitness landscape, and it is often metaphorically called "the amoeba method."

The above two approaches can be effectively used in pairs when one tackles architectural design problems. In terms of pure computational approaches in architectural design, strategic uses of the above two methods in proper conditions will become more effective. For example, the second approach can be applied to problems that have been simplified

to some extent by the first approach. In much problem-solving using a GA, an original population of strings is already represented as genotypes, and these genotypes are encoded from phenotypes that represent actual architectural schemes. This encoding process belongs to the first approach that reduces and simplifies the search space.

7.3 Summary: Comparison of Proposed Systems

Most of the architectural applications introduced in this thesis involve many simplifications of problems, for example, by representing a unit volume of built structures by a voxel. However, most of the primary derivations of structures rely on self-organizing computation or design search methods based on selection procedures, and these computational methods belong to the second approach in the previous discussion – use of the methods with solution-seeking behaviors. This approach is different from brute-force search based on the process of elimination.

Method	Computational Methods (Objectives)	Implemented Algorithms	Subunit	Emergent Structures (derivatives)	Interactions	Behaviors
Circle Packing using a Bubble Meshing Method (Chapter 3)	Self-organizing Computation	Bubble Meshing Method	Circle (Bubble)	Configurations of circles	Concurrent Multiple Interactions	Physics-based (push & pull)
Pedestrian Crossing Simulation (Lane Formations) (Chapter 3)	Self-organizing Computation	Agent-based computation	Pedestrian Agent	Formations by Pedestrians (Lane Formations)	Concurrent Multiple Interactions	
Particle System (Elastic Spring Mass System) (Chapter 4)	Self-organizing Computation Evaluation (Method 1)	Finite Difference Method	Mass Particle linked by Springs	Funicular Structures (Catenary Shapes, Stable Structures)	Concurrent Multiple Interactions	Physics-based (Elastic behaviors of springs, damping forces, and etc.)
Turtle Interpretation of L-system + GA's framework (Chapter 4)	Design Search (Method 2)	L-system Turtle-geometry GA	Voxel (Unit Mass)	Selections based on Fitness Functions	No dynamic interactions	
DLA (Diffusion-limited Aggregation) (Chapter 5)	Growth (Method 3)	DLA Laplacian diffusion equation. Probabilistic model	Voxel (Housing Unit)	Cluster(s) of housing units	Discrete One by One Not multiple interactions	RandomWalk Stop next to the existing clusters
Experiment using Physical Components (Chapter 5)	Adaptation (Method 3)	Multi-dimensional Optimization (Nelder-Meade method)	Panel with light sensor	Panel Configurations	Concurrent Multiple Interactions	
Agent-based simulation of settlement (Chapter 6)	Self-organizing Computation Growth + Adaptation (Method 3)	Agent-based computation	Settler Agents & Surface Patches	Streets & Buildings	Concurrent Multiple Interactions	Agents' Behaviors changes Open loop feedback between agents & environment.

Figure 7.1a – Table showing different strategies and their characteristics

Feedback (+, -)	Randomness (Fluctuations)	Find unknown Goals / solutions? Reconfigurable?	Growth Model?	Advantages, (Pros.)	Limitations, (cons.)
(+) If too close → push (-) If too far apart → pull	Circles' Initial Conditions (YES) Circles' Movements (NO)				
(+) If no obstacles → follow others (-) If collision detected → avoid others	Agents' Initial Conditions (YES) Pedestrian Movements (YES)				
→ Calculate Axial Forces on members → Calculate displacement → Update location	NO Deterministic behaviors				
	GA uses Stochastic selections (Crossover & Mutations)	YES. Turtle can change its path based on fitness values.	Generation by generation, it evolves solutions. However, no continuity among proceeding generations.	Various Fitness functions can be applied.	Limits in Computational speed
If no obstacles → random walk If next to others → stop random walk	Movements of Units (YES & All Random)	YES. YES.	A model can show gradual growth pattern over time.	Represents Gradual Growth patterns	Stochastic model works, but non-deterministic results.
	Initial Condition (YES) Reconfiguration sequences (NO)	Find unknown configurations for new conditions. (However, no-self-reproduction (growth) capability in current design) YES.	Non-growth model. A number of components is fixed for now. Conceptually, can be extended to growth model.	Integrate design tool and physical ideas. Building-scale application requires hardware's technological innovation for more sophistication.	Already similar conceptual precedents in Comp. Sci. area. (But not directly in architectural app.)
Open loop feedback between agents & environment (See chapter 6)	Agents' Initial Conditions (YES) Agents' Behaviors (YES)	Yes	Displays gradual growth patterns	No need to impose existing typological model of cities. Emergent formation.	Hard to gain details without any interpretation by users.

Figure 7.1b – Table showing different strategies and their characteristics

Selection – Design Search method

Among seven applications or experiments in the last three chapters (see Figure 7.1a-b), only the turtle interpretation of L-system + GA system (Chapter 4) belongs to the design search method using selection (method 2 in Chapter 3) and has fewer self-organizing characteristics. This experiment platform allows users to have multiple objectives as evaluation criteria. The objectives include architectural criteria, such as density, structural stability, number of exposed faces of structures, and so on. GA's stochastic selection methods using crossover and mutation are designed to move the population away from local optima, and the system's advantage is that the selections based on a GA can be applied to solve global optimization. However, the system's limitation is that it does not have an ability to produce new objectives that are better suited to given environmental settings.

The physical experiment using a microcontroller (Chapter 5) is classified as an adaptation method (Method 3 in Chapter 3). However an implemented multi-dimensional optimization algorithm, the Nelder-Mead method, shows the characteristics of design search using selection (Method 2 in Chapter 3). Unlike the method based on GAs that can search a fitness landscape globally, this method tends to focus on searching for local optima without the population-based stochastic sampling used in GAs, and the method cannot concurrently search for any other existing alternative potential solutions. This was partly a compromise between calculation speed and real-time physical performance limitations of hardware components. Since the system's platform has a clear definition of subunit as an architectural panel unit with sensing and activation mechanisms, algorithms

that can perform reconfigurations based on self-organizing logics can be implemented for future explorations as the performance and availability of hardware increases.

The following five experiments are considered as a system using self-organizing computation: circle packing using a bubble meshing method (Chapter 3), lane formation by pedestrian agents (Chapter 3), elastic mass particle system (Chapter 4), the DLA experiment (Chapter 5), and agent-based simulation / design of settlement patterns (Chapter 6).

Subunits' Interactions

The DLA experiment is classified as self-organizing computation; however, its type of interaction is not considered as concurrent multiple interactions. Only one subunit (voxel) at a time can move (based on random walk) and interact with the rest of the subunits that are already static. This interaction is discrete one-way sensing and action. In an original DLA model, subunits do not have an ability to reconfigure themselves once they have stabilized. Several approaches introduced in Chapter 5 – swapping and subtraction algorithms – are strategies to improve the DLA system to be a more interchangeable and reconfigurable process. The resulting structures of the DLA are based on linear summation of the individual contributions from these step-by-step accumulations of subunits and show certain geometrical features such as self-similarity. However, whether this discrete process can induce emergence of structures that display qualitatively new properties is not clear. Non-linearity in emergent formation processes is one of the underlying characteristics of self-organization, and multiple interactions are a key attribute for inducing such a characteristic.

Lack of Selection Sequences in Self-organizing Computation

One critical issue with self-organizing computation systems is a lack of selection sequences. All examples of self-organizing computation in this thesis can produce only one result at one time of their simulation runs, and the systems can provide a wider range of solutions if fluctuations are implemented within their behaviors. It is probably more convincing to prepare another layer of evaluation sequence after the generation of self-organizing structures in order to select the best possible structure for a certain condition.

In the case of the DLA experiments, the process produces fluctuations in its resulting structures due to the subunits' random walk behaviors. Since the DLA can produce only one result at a time from one run of the simulation, it is best to provide additional processes to evaluate and select better schemes among all produced structures.

Selection and Self-organization

One of the differences between self-organizing computation and design search based on selection (such as GAs) is whether the methods have a potential to generate new objectives. In the case of selection procedures, normally the systems require fixed fitness or utility functions for selection of better schemes from populations of schemes. It is beneficial to use two approaches in tandem in order to compensate for the shortcomings of each. Figure 7.2 shows a proposed conceptual work flow of two approaches in tandem.

Random or Deterministic Agents' Behaviors

In Chapter 3, randomness is listed as one of the key attributes of self-organizing computation systems. Although randomness is not a necessary condition for such systems

for the emergence of structures, randomness plays an important role in the discovery of new solutions in both systems. (This is also true for design search methods with selection sequences. In the case of a GA, stochastic selection plays a key role in searching unexploited areas in the fitness landscape.)

Circle packing using a bubble meshing method (Chapter 3) and the Elastic Mass Particle system (Chapter 4) do not rely on any types of fluctuations by random behaviors (except the initial configurations of circles – locations and radii – in experiments using a bubble meshing method). In these cases, a single result is deterministically derived. There is no need for an additional selection procedure to gain a result in these cases. The reason for using self-organizing computation for these two cases is that analytical approaches to solve the same problems are not practically feasible.

In the above two cases, the objectives of the experiments are explicit: finding the steady states of structures based on forces in each member, and finding configurations of circles that fit inside a larger circle. Locally defined behaviors of subunits are carefully chosen to satisfy targeted objectives, or empirically these behaviors are already known.

On the other hand, in the case of pedestrian agents, we only provide behaviors of agents, but objectives of collective agents are not clearly known or defined at the outset. Lanes formed by pedestrian agents are results induced from agents' locally defined behaviors. Of course, collision avoidance and repelling behaviors of agents already indirectly imply smooth and steady flows of agents as a collective objective of pedestrian groups; however, for some other scenarios, we may find an unpredicted objective as a byproduct

of emergent structures that are induced from locally defined behaviors of agents. An implemented set of agents' behaviors may produce globally functional structures that indicate unpredicted objectives.

**Platform development of
Self-organizing Computation:**

Define environment and boundary conditions of a system.

Define subunits of a system.

Define behaviors of subunits.

- **Interaction Type**
- **Feedback**
- **Fluctuations (randomness)**

(If one already has clear objectives, objectives need to be expressed through behaviors of subunits. This mapping is not trivial for some cases.)

Define initial condition of a system.

States of subunits

States of environments

Initial set of parameter values (if parameters exist)

Run a system.

→ Results

Fitness landscape for Selection

Selection procedure (Design Search method)

Define Solution Space (Initial conditions and Results from SOC)

Define objectives for selection (fitness functions)

Define selection sequence methods (GA, etc.)

→ Optimal results

Figure 7.2 – Flow chart summarizing a platform development of self-organizing computation

Open Loop Feedback System

The application in Chapter 6, agent-based simulation/design of settlement patterns, is the only system that can change subunits' behaviors over time based on stimuli from the changing environment. All other systems' subunits that were previously introduced have fixed behaviors, and they do not update themselves over time. This co-evolutionary process between agents and environments is known to exist in many self-organizing systems in nature, as introduced in Chapter 2. There are positive and negative feedbacks inside such systems. In the case of this system from Chapter 6, trails that are frequently used by agents become attractors for agents in their neighborhoods. Consequently, the frequency of these trails' usage is amplified. This is a typical example of positive feedback. The trails that have not been used regularly by agents start to lose their value for attraction. The value for attraction – intensity of traffic – is directly related to the visibility of the trails, so that the latter trails will eventually fade away. The positive and negative feedbacks work as a pair to produce organic gradual transformations of street patterns such as bundling and branching in the system.

Adaptation

This system from Chapter 6 has the key ingredients of self-organizing systems introduced in Chapter 3; open loop feedback, multiple interactions, and randomness. The system's objectives are also not as obvious as cases of circle packing or simulation of pedestrian crossings, and the system has more generative aspects than the others. Primary inputs of the system are essential information about the site conditions, such as topography, and basic agents' behaviors. These original implementations alone may not be sufficient to

produce explicit meanings for objectives of the system. Environmental growth of the system and agents' behavioral changes lead the entire system to gradually adapt itself to emerging states of the system. "Developing shortest path patterns for agents to travel around cities," "configuring economical path patterns for construction," "creating paths that have least elevation changes for agents' trips," or "building allocations that can maintain functional traffic patterns," are several interpreted objectives from the results of the system with various different parameter settings. These objectives are ex post interpretations of the results and are not provided directly as initial requirements or goals for the system. During the course of the system's run, these changes toward satisfaction of certain objectives emerge as a result of the system's ability to adjust its response to stimuli according to the state of the environment.

7.4 Application Areas That Benefit from the Advantages of Self-organizing Computation

Growth Model

In conclusion, there are several advantages of self-organizing computation in development of generative design systems in architecture. Firstly, the most obvious advantage is that self-organizing computation can represent and generate growth processes over time. This means that the approach can design a system in transition. Recently, design of architecture is not all about providing a single design instance static in time. Many buildings and urban developments are required to have flexibility to accommodate new unexpected usage patterns and new spatial demands for emerging architectural programs. Extensibility of the system is needed for new generations of

buildings, and many buildings are expected to be planned on the basis of an open-planning approach from now on.

Performance-based Design

Another advantage associated with its ability to accommodate growth, is its suitability for performance-based applications. In Chapter 2, several existing models of design directed by multiple individuals such as Kowloon Walled City were introduced, and they were all manually executed without use of any computational systems. As results, they produced many malfunctioning spatial components within their complexes – for example, deficient lighting and circulation inside the dungeon-like structures. Self-organizing computation is the computational approach that can bring more accuracy and precision based on analytical measures of structures' performances without loss of their decentralized adaptable characteristics. For example, the DLA experiments in Chapter 4 use probability fields for the lighting conditions to quantify values associated with the structures. Potentials we found from spontaneous growth such as Kowloon's adaptability for rapidly growing populations can be greatly enhanced by computational simulations based on self-organizing computation.

Objectives in Transition

In professional practice in architecture, in most cases, requirements for certain programs and sizes of architectural space are given from the outset, based on clients' budgets and code regulations. Objectives for a given space are explicit information in many cases of conventional architectural design work. When these requirements are not completely

given by the design problems' frameworks, the objectives themselves need to be generated from primary information about sites such as landform, environmental conditions, sites' boundary conditions, and so on. As I mentioned in the introductory chapter, such conditions have begun to be found among large-scale complex buildings that are undergoing constant changes of their usage.

One of the byproducts of a growth model is the ability to produce new objectives based on the model and its environment's coevolutionary development processes. Discovering unknown design objectives (in a sense, the emergence of new architectural programs) is one unique characteristic that is potentially applicable to planning and strategic development of architectural projects in earlier stages. When sufficient information about the proposed buildings and site is not available, the system can become an active provider of potential schemes. The parameters that govern the future changes of the system – its growth – should be adjusted based on the comparisons of simulation results that are selected by the developers of the planning and designs. Consequently, these parameter values will become (formerly unknown) factors for projects' initial conditions.

Decentralized Systems and System Control

Self-organizing systems that consist of subunits and their interactions naturally possess a decentralized character. This decentralized character found in self-organizing computation systems is obviously advantageous for developing operation systems for buildings with some types of adjustable functionalities. Advantages of decentralized systems are their distributedness and robustness, where failures of some parts of a system do not always lead to a failure of the entire system.

Typical examples of such building components are operable windows for lighting and air ventilation, mechanical shading devices such as louvers and canopies, movable partitions, and so on. If the controls of such devices are distributed, local failures do not cause a failure of the entire system. The architectural panel system using physical components introduced in Chapter 5 is a good example of this characteristic's potential. Since sensing and actuation mechanisms are locally embedded inside each individual panel, the system will still try to function even if one or two components stop working. Furthermore, there are potentials to build evolvable systems that can learn from sensors and actuators through open feedback loops. A distributed component, as an agent, can learn new appropriate behaviors based on feedback from the actual physical environment rather than always following initially predefined instructions of the system. This is similar to the implementation of open loop feedback for agents in the experiments in Chapter 6.

In the case of extensible growth models, discussion goes beyond the simple operations of buildings with fixed components and functionalities. For example, in the case of the aforementioned physical panel system from Chapter 5, panels will no longer be four fixed panels; instead they will have particular joint systems to link and reconfigure to satisfy unknown objectives. This ability to reconfigure is the case where the concept of growth is applied to conventional systems, and this adaptability for growth models separates self-organizing computation from many other existing design computing systems.

7.5 Remarks and Implications

Explicit or Gradient Representations

The application introduced in Chapter 6 uses agents which have solution-seeking behaviors based on self-organizing computation. They are able to come up with street network systems without having any types of recipe, template, or typology for street patterns. Representations and derivations of their results all rely on rasterized gradients that indicate degrees of traffic frequency. This representational style has been consistently used throughout the simulation since the style enables the representation of the gradual processes of street pattern generation over time.

The same results can be treated and represented as topological graphs using vectorized graphs; however, this binary, on-or-off, type of representation is not suited for displaying gradual growth in a spatiotemporal manner. But transformation sequences from one state to another can be well visualized through the color gradient map (Figure 7.3).

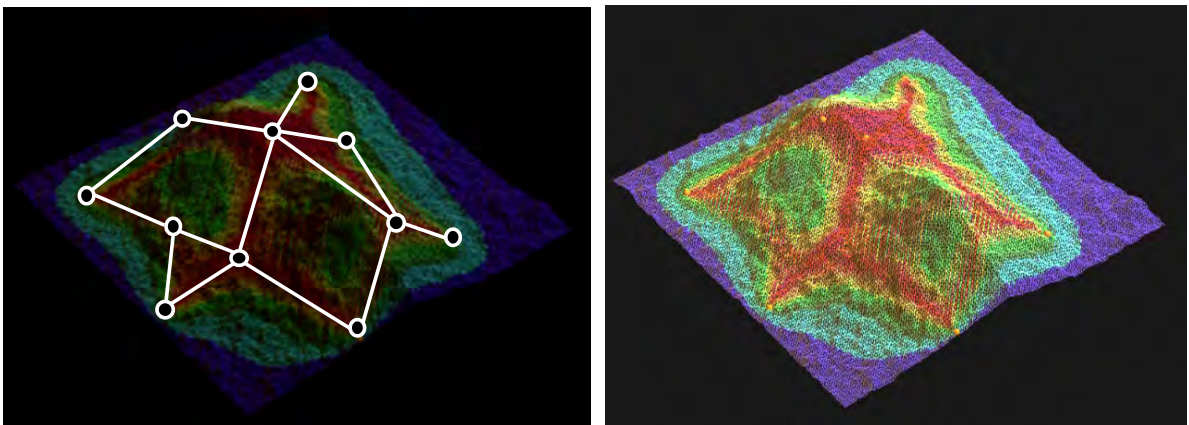


Figure 7.3 – Vectorized representation and rasterized gradients representation

A light quantum (a photon) can be interpreted as both a particle and a wave. Similarly, the same dualistic interpretation can be applied to many design phenomena dealing with physical morphology. Problems of street generation can be reduced to problems of mathematical graphs, and one can gain many benefits from analytical aspects of graph theory by taking this direction. The traveling path problems famously represented by the Königsberg Bridge problem are cases where the graph approach becomes more advantageous (Figure 7.3).

One of the limitations in current CAD systems is the lack of dualistic interpretation and representations. A discrete vector-based representation alone may not be able to capture many biological growth phenomena in an efficient and dynamic manner, and a generative use of such systems as a representation of results from self-organizing phenomena may have some limitations. A one-sided point of view may cause some deficiencies. Our available view of the world simply does not allow us to think simultaneously in two ways. It may be advisable to switch between two interpretations to take advantage of both approaches, according to a given problem's nature.

Degree of Reliance on External Knowledge

One of the unique characteristics of self-organizing computation is its non-reliance on any external knowledge. It is a self-sufficient autonomous process and requires no recipe, template, or typology of formal instructions for the emergence of spatial structures. In chapter 6, all the street patterns derived were autonomously driven by the behaviors of agents. Even some of the agents' behaviors are stimulated by environmental changes and indirectly derived. The system has absolutely no knowledge about whether the

intersections have a radial or grid pattern, or rural or urban settings. This is the major difference from the aforementioned L-system and shape grammar-based approaches. Imposing existing design patterns or transformation sequences is beneficial when one wants to efficiently derive what appear to be the subjects of our recognitions. However, reliance on a pre-existing template might preclude the possibility of discovering what original inputs naturally turn into. In this conceptual experiment, any imposition of knowledge from outside of the system has been thoroughly rejected from the process. However, subsequently, I still recognized some of the street configurations or settlement patterns that are familiar to our perceptions. What is separating the results from direct paths, detours, or minimal ways is a simple set of parameters. Instead of providing predefined design templates such as grid, radial, or branching patterns, sets of parameter values that generate agents' behaviors are driving the resulting configurations. In this way, the inherent characteristics of the resulting configurations are traceable back to several parameter values that govern the behavior of the system, and certain sets of parameters that lead to characteristics similar to existing urban phenomena can be studied. The goal of the experiment is to derive forms from behaviors instead of supplying a formal knowledge of design patterns at the outset.

Induction Methods

The early phase of the experiments in Chapter 6 has captured patterns seen in some existing settlements. The results of the simulation are induced by a repertoire of a few behaviors implemented in agents, and some of the behaviors spontaneously emerged from a system's growth itself. This characteristic of the system implies that some

tendencies found in human design activities are, to some extent, captured or approximated by a few repertoires of major behaviors by agents.

Of course, resulting configurations based on decisions by groups of humans are often results of complex negotiations and superimpositions of multiple results over time, and complete descriptions of these processes may not easily be reduced to a simple set of parameters. However, one of the main differences from some existing city generation software is that the system does not impose specific design templates. With a few primary inputs related to the site's geographical information as initial conditions, the system can spontaneously produce all design components using self-organizing computation.

Finding an underlying principle for generative processes of man-made design is an under-explored area of investigation. This thesis studied the generative approach and further explored the potential applications (by actively using computational interpretations) for a generative approach. Whether the formation logics of man-made structures can be reduced to a simple form of certain principles is a question. A recent paper by Schmidt and Lipson (2009) presents a computational experiment to distill natural laws of physical systems, such as harmonic oscillators, from motion-tracking data captured from various physical systems. After Einstein's theory of general relativity, we all know that even a Newtonian interpretation of physics is not the complete description of our physical systems. Yet it is widely accepted as a reliable means to describe dynamics of our physical world. Furthermore, economists' efforts to mathematically model human decisions and to study their consequences face a similar challenge, as we are not perfectly

sure artificial systems' behaviors can be reduced to analytical descriptions based on mathematics.

Modeling and Designing (Simulation and Generation)

Modeling is very different from designing a generative design application in architecture. Creating new design systems that can generate adaptive, flexible, and robust design solutions is the main goal of the thesis. As a first step toward developing generative systems for new instances of design, the mechanisms of systems that can produce and simulate existing instances were studied.

In experiments from Chapter 6, an earlier phase of application run has displayed characteristics of modeling and simulation more than the later phases of them. The results from the earlier phases were compared with actual conditions of the proposed site, and the later phase shows the hypothetical results where incoming population and traffic intensities of the site area continue to increase. The current site, the San Miniato area, has a moderately settled, relatively rural condition, yet some of the results from later phases show developments close to the typical density of metropolitan-class cities. The simulation and prediction phases are seamlessly joined in the case of these experiments. Various parameters that govern the agents' local behaviors changed the later patterns of the cities' developments. One can decide which schemes to select by finding which schemes induce global behavioral patterns to emerge that one prefers agents to enact.

Future Research

Computational design application tools in architecture are on the brink of transition from being mere analytical tools to becoming more creative tools that can induce emergence of new solutions, or at least serve as “co-evolvers” of design solutions for humans. This transition in the role of computational tools will have a big impact on our design communities and will pose a question about what the actual roles of human design experts in the field are. Further, the rise of collective design interface platforms may change our current value systems in architecture and design.

This thesis has investigated computational strategies that could advance this transition and proposed generative approaches through conceptual experiments. An obvious next step is to find real-life scenarios to which this approach is applicable and develop feasible systems that go beyond conceptual desktop experiments to become practical off-the-shelf solutions. More specifically, experiments in chapter 6 primarily involved landscape—urban design applications based on agents’ movements constrained within two-dimensional landform surfaces. A technically challenging next step is to extend this system’s emergent characteristics to more general design settings and purposes and throughout different scales.

A Question about Application Scales and Areas

In human designs, as stated in chapter 2, we have witnessed more examples of emergent phenomena from urban-scale city formations rather than building-scale applications. The choice of the application scale for the experiments in the last chapter was also urban.

However, it is not my intention to say that applications in urban-scale settings are more suitable for methods inspired by self-organizing computation. Normally, urban growth occurs through spontaneous processes over many years, and metabolic rates and demands for adaptation can tolerate the time that it takes for humans to construct and alter structures in urban scenarios. In the case of individual buildings, or clusters of a few, the metabolic rate and demand for buildings to change can occur, for some cases, within a shorter range of time. Even within the span of a day, some buildings face demands for changes (think of responding to changing external light and temperature), and buildings' components therefore need more active mechanisms for adaptation, rather than requiring years for reconstruction, alterations, and demolition. Physical and technical demands are obviously higher for these scales of applications than for complete urban scenarios. This reality explains the current interest in many self-reconfigurable systems among computer scientists and some architectural thinkers. In addition, as I have reviewed in chapter 2, a relatively low-cost and primitive housing structure, such as Kowloon Walled City, displays a more spontaneous growth process due to its relatively simple and easy construction methods, and this clearly illustrates a tendency. This fact leads us to have more difficulties implementing self-organizing logics at a finer physical scale (for aggregations of subunit components) and with a shorter time scale for changes. Achieving more active and rapid reconfiguration at a finer scale is also an immense mechanical challenge with our currently available technology. As a result, we seemingly find more opportunities in urban-scale applications with this logic.

However, as regards constructing design strategies spatiotemporally, incorporation of self-organizing methods is effective regardless of the scale of applications. Physical and

tectonic demands will vary, depending on a time scale for growth of systems, but primary conceptual design directions can be enriched by adaptation of methods inspired by self-organization. Rather than imposing recipes or templates, this thesis has looked at inductive design methods based on self-organizing computation. The results have shown unmistakably that the behavioral implementation can induce configurations like those found in man-made systems. Further active uses of this knowledge to generate instances from unknown conditions can confidently be anticipated.

List of Figures

This appendix lists the credits for the figures contained within the thesis. All figures not listed below were produced by the author.

- 1.1 http://images.travelpod.com/users/ditchthecube/5.1265552228.1_housing.jpg
- 2.2 http://en.wikipedia.org/wiki/Image:Termite_Cathedral_DSC03570.jpg
- 2.3 Theraulaz and Bonabeau, (1995a).
- 2.7 Lipson, (2005).
- 2.8 Yatsuka, (1997).
- 2.9 Shiokawa, et. al, (2000).
- 2.10 Nomura et al. 2006
- 2.11 Lipson, (2000).
- 2.12 Bowdon et al, (1997).
- 2.13 Left: <http://reprap.org/bin/view/Main/WebHome> (RepRap, 2010).
Right: Malone and Lipson, (2006).
Bottom: Malone and Lipson, (2005).
- 2.15 Lambot, (1999).
- 2.16a *ibid.*
- 2.16b *ibid.*
- 2.17a *ibid.*
- 2.17b *ibid.*
- 2.19 Gassel, (1979).
- 2.20 Barros and Sobreira, (2002).
- 2.21 *ibid.*
- 2.22 Muller and Parish, (2001).
- 2.23 *ibid.*
- 2.24a Alexander and Manheim, (1962).
- 2.24b *ibid.*

- 3.1 Katoh, (1980).
- 3.12 Top-left: <http://www.auto desk.com> (AutoDesk co. ltd.)
Top-middle: <http://www.hitachi-kokusai.co.jp/goyo/html/kaiseiki.html>
Top-right & Bottom-left: <http://forum8.co.jp> (Forum8 co. ltd.)
- 3.16 Kirchner, Nishinari, and Schadschneider, (2002).
- 3.17 Left: <http://www.optimalsolutions.us>.
Middle & right: <http://www.fe-design.de> (Fe-Design GmbH, Germany).
- 3.19 Nowak, (2006).
- 3.20 Brand, (1997).

- 4.3.1 Left and Middle: Sanders, (2000).

- 5.1 <http://irishabroad.blogspot.com>
- 5.2 Costa, (1977).
- 5.3 Top-left and bottom: GoogleEarth, (2010)
Top-right: <http://www.panoramio.com/photo/4720638>
- 5.4 Mykonos
- 5.x Left: Fanelli, (1990).
- 5.x Ferguson, et al. (1990); and Brant, (1994).

References:

- Abelson H. and diSessa A. (1982). *Turtle geometry*. MIT Press, Cambridge.
- Alexander, Christopher, and Marvin L. Manheim (1962). *The Use of Diagrams in Highway Route Location*. Civil Engineering Systems Laboratory, Massachusetts Institute of Technology, 1962. Research Report R62-3.
- Alexander, Christopher, (1965). A city is not a tree, *Architectural Forum*, vol. 122, No 1, April 1965, pp 58-62 (Part I), vol. 122, No 2, May 1965, pp 58-62 (Part II)
- Alexander, Christopher (1966). From a set of forces to a form." In Gyorgy Kepes (Ed.), *The Man-Made Object* (pp. 96-107). New York: George Braziller.
- Alexander, Christopher (1964). Notes on the Synthesis of Form. Cambridge, Harvard University Press.
- ARES Project. (2007). [Online]. Available: <http://www.ares-nest.org/> The ARES modular microrobotic system in various configuration in the GI tract. ARES images courtesy Paolo Corradi, Scuola Superiore Sant'Anna, Italy.
- Arduino, (2010). <http://www.arduino.cc> [Accessed 6-25-2009].
- Barros, J. and Sobreira, F., (2002), City of Slums: self-organization across scales. Centre for Advanced Spatial Analysis – CASA, University College London, Working Papers.
- Batty, M. and Longley, P. (1994). *Fractal Cities: A Geometry of Form and Function*, Academic Press, San Diego, CA and London.
- Batty, M. (2006). *Cities and Complexity: Understanding Cities with Cellular Automata, Agent-based models, and Fractals*, Cambridge, MA. MIT Press.
- Benenson, I. and Torrens, P. (2004). *Geosimulation: Automata-based modeling of urban phenomena*. :Wiley.
- Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). *Swarm Intelligence: from natural to artificial intelligence*. Oxford University Press. New York.
- Bonabeau, E., Guérin, S., Snyers, D., Kuntz, P., Theraulaz, G. (2000). Three-dimensional architectures grown by simple 'stigmergic' agents, *Biosystems*, 2000 - Volume 56, Issue 1, March 2000, Pages 13-32, Elsevier.
- Bongard J., Zykov V., Lipson H. (2006), Resilient Machines Through Continuous Self-Modeling, *Science* Vol. 314. no. 5802, pp. 1118 - 1121.
- Boulding, Kenneth E. (1996). *Evolution, order, and complexity*. London; New York: Routledge.
- Bowden, N. et al, (1997). Self-Assembly of Mesoscale Objects into Ordered Two-Dimensional Arrays, *Science* 11 April 1997 276: 233-235

- Brand, S. (1994). *How Buildings Learn: What Happens After They're Built*. New York: Viking.
- Caldas, Luisa G. and Norford, Leslie K. (1999). A Genetic Algorithm Tool for Design Optimization, Media and Design Process [ACADIA '99 / ISBN 1-880250-08-X] Salt Lake City 29-31 October 1999, pp. 260-271
- Caldas, Luisa Gama and Norford, Leslie K. (2002). A design optimization tool based on a genetic algorithm, *Automation in Construction* 11 (2) (2002) pp. 173-184
- Camazine, Deneubourg, Franks, Sneyd, Theraulaz, Bonabeau, (2002). *Self-organization in Biological Systems*, Princeton University Press. Princeton, New Jersey.
- Costa, Paolo, (1977). *Yemen, land of builders*, London, Academy Editions
- Damlūji, Salmá Samar, (1992). *The valley of mud brick architecture: Shibām, Tarīm & Wādī Hadramūt: ancient to contemporary design*, Reading, UK. Garnet Publishing Limited.
- Daniela Rus, Zack Butler, Keith Kotay, Marsette Vona, (2002), Self-reconfiguring robots, *Communications of the ACM*, Volume 45, Issue 3, March 2002.
- Dawkins R. (1976). *Selfish Gene*, Oxford; New York: Oxford University Press.
- Deneubourg, J. *et al.* (1990). The Dynamics of Collective Sorting: Robot-Like Ants and Ant-like Robots, *Simulation of Adaptive Behavior: from animals to animats*, Cambridge, MA, MIT Press.
- Di Caro, G., and Dorigo, M. (1997). AntNet: A Mobile Agents Approach to Adaptive Routing, *Technical Report IRIDIA/97-12*. Universite Libre de Bruxelles, Belgium, 1997
- Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milan, Italy.
- Duarte, J. P. *et al.*, (2007). Unveiling the structure of the Marrakech Medina: A shape grammar and an interpreter for generating urban form, *artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Cambridge University Press.
- Dudnik, Elliott E. and Schachtel, Wayne, (1974). A computer aided land use study technique Annual ACM IEEE Design Automation Conference archive, *Proceedings of the 11th Design Automation Workshop table of contents*, Pages: 237 – 247, IEEE Press Piscataway, NJ.
- Fab@Home. (2010). <http://www.fabathome.org/> [Accessed April 25, 2010].
- Fanelli, G. (1990). *Toscana*, Francesco Trivisonno Firenze : Cantini.
- Kepes, G. (1966). *The Man-made Object*, New York, G. Braziller.
- FE-Design GmbH, (2010). [software company]. <http://www.fe-design.com/> [accessed April 25, 2010].
- Fonseca, C. M., and Fleming, P. J., (1995). “Multi-Objective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restrictions,” *Proc. of 1st Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications*, IEE Conf. Pub, 414, pp. 45–52.

- Franpton, K., (1992). *Modern Architecture: A Critical History*. New York, NY. Thames and Hudson.
- Ferguson, T. J., Mills, B. j., and Seciwa, C. (1990). Contemporary Zuni Architecture and Society, *Pueblo Style and Regional Architecture*, New York: Van Nostrand Reinhold. pp103-121.
- Focas, C. et al. (1998). *The four world cities transport study*, London: Stationery Office.
- Gilbert, N. and Troitzsch, K. G. (2005). *Simulation for the Social Scientist*. Open University Press.
- Hall, E. T. (1969). *The Hidden Dimension*, Garden City, New York: Doubleday,
- Haken, H. (1983). *Synergetics*, Berlin: Springer-Verlag.
- Helbing, D. and Molnár, P. (1995). Social force model for pedestrian dynamics *Phys. Rev. E* 51 4282-6.
- Helbing, D. (1997). Self-organization of observed collective behavior of pedestrian *Verkehrsdynamik*, Berlin, Germany: Springer.
- Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating dynamical features of escape panic, *Nature* 407, 487
- Helbing, D., Farkas, I. J., and Bolay, K. (2001). Environment and Planning B, 2001, Self-organizing pedestrian movement, *Environment and Planning B: Planning and Design 2001*, volume 28, pages 361 ^ 383.
- Holland, J. (1992). *Genetic Algorithms*, Scientific America, July 1992.
- Hornsby, Gregory S. and P. Jordan. B. (2001). The Advantages of generative Grammatical Encodings for Physical Design, *Congress on Evolutionary Computation*.
- Hornby, Gregory S. (2003). *Generative Representations for Evolutionary Design Automation*. Ph.D. Dissertation, Brandeis University Dept. of Computer Science.
- Isozaki, A. (1967). Time Factor on Architectural Planning (Features Symposium), *Kenchiku-zasshi*, Architectural Institute of Japan, vol. 82(986), 1967-09-20, pp608-609.
- Isozaki, A. (1972). *Kuukannhe*, Bijyutu-shuppan-sha, 1971.
- Kaandorp, J. (1994). "Fractal Modeling: growth and Form in Biology", Springer-Verlag.
- Kahneman, D. and Tversky, A. (1979). Prospect Theory: An Analysis of Decision under Risk", *Econometrica*, XLVII (1979), 263-291.
- Katoh, I. et al. (1980) Characteristics of lane formation, *Nihon Kenchikugakkai Ronbun houkokushu*, No289, p121
- Kauffman, Stuart, (1993). *Origin of Order*, New York: Oxford University Press.

- Kauffman, Stuart, (1995). *At Home in the Universe: the Search for Laws of Self-organization and Complexity*, Oxford; New York: Oxford University Press.
- Kauffman, Stuart, (2000). *Investigations*, Oxford; New York: Oxford University Press.
- Kauffman, Stuart, (2008). *Reinventing the sacred: a new view of science, reason and religion*, New York: Basic Books.
- Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. *Proc. IEEE International Conf. on Neural Networks (Perth, Australia)*, IEEE Service Center, Piscataway, NJ, (in press).
- Kraitchik, M. §8.4.1 in *Mathematical Recreations*. New York: W. W. Norton, pp. 209-211, 1942.
- Kirchner, Ansgar, Nishinari, Katsuhiro, and Schadschneider, Andreas, (2002). Friction effects and clogging in a cellular automaton model for pedestrian dynamics, *Statistical Mechanics* Institut für Theoretische Physik, Universität zu Köln D-50937 Köln, Germany, (Dated: February 1, 2008).
- Konaka, Abdullah, Coitb, David W. and Smith, Alice E. (2006). Multi-objective optimization using genetic algorithms: A tutorial, *Reliability Engineering and System Safety* 91, 992–1007
- Kroll, L. (1987). *Architecture of Complexity*, Cambridge, MA. MIT Press.
- Lambot, I. and Girard, G. (1999). *City of Darkness: Life in Kowloon City*, Watermark pub.
- Larson, C. R. and Odoni, R. A. (1981). *Urban Operations Research: Logistical and Transportation Planning Methods*. New Jersey: Prentice-Hall.
- Le Bon, G. (1896). *The Crowd: A Study of the Popular Mind*.
- Lindenmayer A. and Prusinkiewicz P. (1990). *The Algorithmic Beauty of Plants*, Springer-Verlag, New York.
- Lipson, H., Pollack J. B., (2000). Automatic Design and Manufacture of Artificial Lifeforms, *Nature* 406, pp. 974-978.
- Lipson H. et al, (2005) Self-reproducing machines, *Nature* Vol. 435 No. 7038
- Malone E., Lipson H., (2005) Freeform Fabrication of Ionomeric Polymer-Metal Composite Actuators, *Proceedings of the 16th Solid Freeform Fabrication Symposium*, Austin TX, Aug 2005, pp. 484-502
- Malone E., Lipson H., (2006) Fab@Home: The Personal Desktop Fabricator Kit, *Proceedings of the 17th Solid Freeform Fabrication Symposium*, Austin TX.
- Maréchaux, Pascal, (1998). *Yemen*, Phebus Editions, Paris.
- Mataric, M. (1995) Designing and Understanding Adaptive Group Behavior, *Adaptive Behavior* 4:1, Dec, pp. 51-80.
- Maynard Smith, J. (1982) *Evolution and the Theory of Games*. Cambridge University Press.

- Merks, R.M.H., (2003). *Branching growth in stony corals: a modelling approach*. Ph.D. Thesis, University of Amsterdam.
- McNeel, R. (2007). RHINOCEROS®: NURBS modeling for Windows [computer software]. <http://www.rhino3d.com>.
- Microsoft Corporation, (2010). [http://msdn.microsoft.com/library/z1zx9t92\(VS.100\).aspx](http://msdn.microsoft.com/library/z1zx9t92(VS.100).aspx) description of Visual C# programming language. [Accessed April 25, 2010].
- Minsky, Marvin and Papert, Seymour, (1972). (2nd edition with corrections, first edition 1969) *Perceptrons: An Introduction to Computational Geometry*, The MIT Press, Cambridge MA
- Mitsui, Ohsaki, Ohmori, Tagawa, Homma, (2004). *Heuristic Methods for Optimization of Structural Systems*, Corona publishing co., ltd. Tokyo Japan.
- Muller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). *Procedural Modeling of Buildings. In Proceedings of ACM SIGGRAPH 2006 / ACM Transactions on Graphics*.
- Murata, S. (2006). Modular Structure Assembly Using Blackboard Path Planning System. *ISARC2006*, Tokyo Institute of Technology, Interdisciplinary Graduate School of Science and Engineering.
- Narahara T. and Terzidis, K. (2006). Multiple-constraint Genetic Algorithm in Housing Design, *International Conference, Synthetic Landscapes / Digital Exchange*, Louisville (USA) 12-15 October.
- Nicolis, G., and Prigogine, I. (1977). *Self-Organization in Non-Equilibrium Systems*. New York, NY: Wiley & Sun.
- Nihon-kenchiku-Gakkai (1994). *Kenchiku shiryou shuusei*. Tokyo, Japan.: Maruzen.
- Nowak, M. A. and May, R. M. (1992) Evolutionary Games and Spatial Chaos, *Nature*, 359(6398), 29 October, pp. 826-829.
- Nowak, M. A. (2006). *Evolutionary Dynamics: exploring the equations of life*, The Belknap press of Harvard University Press.
- Ohuchi, Yamamoto, Kawamura, (2002). *Theory and Application of Multi-agent Systems – Computing Paradigm from Complex System Engineering*, Corona publishing co., ltd. Tokyo Japan.
- OpenGL, (2010). http://www.opengl.org/documentation/current_version/ The Industry's Foundation for High Performance Graphics. [Accessed April 25, 2010].
- Open Toolkit Library (OpenTK), (2008) <http://www.opentk.com/> [Accessed April 25, 2010].
- Parish, Y. I. H., and Muller, P. (2001). Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 301–308.
- Portugali, J. (2000), *Self-organization and the City*, London, Springer-Verlag.

- Portugali, J. (2006). *Complex artificial environments: simulation, cognition and VR in the study and planning of cities*, Berlin: Springer.
- Prigogine, I. and Stengers, I (1984). *Order Out of Chaos*, Bantam.
- Prokopenko et al, In John Fulcher (ed.) (2006), Self-Organizing Impact Sensing Networks in Robust Aerospace Vehicles, *Advances in Applied Artificial Intelligence*, 186-233, Idea Group, 2006.
- Quarantelli, E. (1954). The Nature and Conditions of Panic. *The American Journal of Sociology*, 60(3): 267-275, 1954.
- Radford, Antony D. and Gero, John, S. (1980). On optimization in computer aided architectural design, *Building and Environment*, Volume 15, Issue 2, 1980, Pages 73-80.
- Reis, George E. (1975). Dense Packing of Equal Circles within a Circle, *Mathematics Magazine*, Vol. 48, No. 1 (Jan., 1975), pp. 33-37, Published by: Mathematical Association of America.
- RepRap. (2010). http://reprap.org/wiki/Main_Page, [Accessed April 25, 2010].
- Revit architecture, (2010). © Copyright 2010 Autodesk, Inc. [computer software]. <http://www/Autodesk.com/RevitArchitecture>. [accessed April 25, 2010].
- Sakai, K. (2005). *Dynamics Animation using OpenGL*, Morikita publishing co., ltd. Tokyo Japan.
- Sakai, K. (2008). *3D Computer Graphics and Animation using OpenGL*, Morikita publishing co., ltd. Tokyo Japan.
- Sander, L. M. (2000). *Contemporary Physics*, Volume 41, Issue 4 July 2000, pages 203 – 218
- Schaffer, J. D., (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *Proc. of Ist. Int. Conf. on Genetic Algorithms*, J. J. Grefenstette ~ed., Lawrence Erlbaum, pp. 93–100.
- Schaur, Eda, (1991). Non-planned settlements: Characteristics features – path system, surface subdivision, Editor Frei Otto, *volume 39 of Institute for lightweight structures (IL)*, University of Stuttgart, Germany.
- Schmidt M., Lipson H. (2009) Distilling Free-Form Natural Laws from Experimental Data, *Science*, Vol. 324, no. 5923, pp. 81 – 85.
- Schoonderwoerd, R., Holland, O., Bruten, J., and Rothkrants. L., (1996). Ant-Based Load Balancing in Telecommunications Networks, *Adapt. Behav.* **5** pp. 169-207
- Schweitzer, F. and Schimansky-Geier. L. (1994). Clustering of “active walkers” in a two component system. *Physica A* 206, 359–379.
- Shiokawa, Takashi. et al. (2000). Automated construction system for high-rise reinforced concrete buildings, *Automation in Construction* 9_2000. pp. 229–250
- Shiokawa, Takashi. (2004). Analysis of the improvement process and evaluation by the construction process and the man-hour: A study on development and application of a building

- automation construction system: Part 1, *Journal of Structural and Construction Engineering* (Transactions of AIJ), vol.; no.582; pp15-22
- Simon, H. A. (1969). *The sciences of the artificial*, Cambridge, Massachusetts: MIT Press.
- Sommer, R. (1969). *Personal Space, The behavioral basis of design*. Englewood Cliffs, New Jersey: Prentice Hall Inc.
- Sugihara, K. (2006). *A mathematical Principle of Geometry and Dynamics — Geometry as a means to explore Engineering*, Tokyo, Japan, Tokyo University Press.
- Tamaki, H., Kita, H., and Kobayashi, S. (1996). Multi-Objective Optimization by Genetic Algorithms: A Review, *Proc. of 1996 IEEE Int. Conf. on Evolutionary Computation*, pp. 517–522.
- Terzidis, K. (2006). *Algorithmic Architecture*. Burlington, MA. Architectural Press.
- Terzidis, K. (2009). *Algorithms for Visual Design Using the Processing Language*, Indianapolis: Wiley Publishing, Inc.
- Theraulaz, G. and Bonabeau, E. (1995a). Coordination in Distributed Building, *Science*, 269: pp. 686-688.
- Theraulaz, G. and Bonabeau, E. (1995b). Modeling the collective building of complex architectures in social insects with lattice swarms, *Journal of Theoretical Biology*.
- Turing, A. (1952). The chemical basis for morphogenesis. *Philosophical Transactions of the Royal Society of London* 237:37-72.
- Turner, A and A. Penn (2002). Encoding natural movement as an agent-based system: an investigation into human pedestrian behaviors in the built environment. *Environment and Planning B: Planning and Design*. 473-490.
- Tversky, A. & Kahneman, D. (1991). Loss Aversion in Riskless Choice: A Reference Dependent Model. *Quarterly Journal of Economics* 106, 1039-1061.
- Varanda, Fernando, (1982), *Art of building in Yemen*, Cambridge, MA. MIT Press.
- Warneke, B. et al, (2001) Smart Dust: communicating with a cubic-millimeter computer, *Computer IEEE* On page(s): 44-51, Volume: 34, Issue 1, Jan 2001
- Werfel, J. and Nagpal, R. (2006). Extended Stigmergy in Collective Construction, *IEEE Intelligent Systems* 21(2): 20-28.
- Weber, B., Müller, P., Wonka, P., and Gross, M. (2009). Interactive Geometric Simulation of 4D Cities, *EUROGRAPHICS 2009 / P. Dutré and M. Stamminger (Guest Editors) Volume 28, Number 2*.
- Werner, B. (1997). *Mies van der Rohe*. Berlin, Germany: Birkhauser Verlag.
- Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Witten, Jr, T.A. and Sander, L. M. (1981) Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon, *Physical Review Letters*.

Yatsuka, H. and Yoshimatsu, H. (1997). *Metabolism: senkyūhyaku rokujū-nendai Nihon no kenchiku avangyarudo*, Tokyo: Inakkusu Shuppan.

Relevant Publications by the Author

- Narahara T. (2010). Design for Constant Change: Adaptable Growth Model for Architecture, *International Journal of Architectural Computing (IJAC)*, IJAC 8-1.
- Narahara T. et al. (2009a). Book Chapter in *Computational Constructs: Architectural Design, Logic, and Theory*, (This publication is coordinated between MIT Design and Computation Group and Digital Architecture Group of World Association of Chinese Architects (WACA) and is funded by WACA.) The China Architecture and Building Press, China.
- with Yoshihiro Kobayashi, Kostas Terzidis, et. al. (2009b). World8: International Working Group for New Virtual Reality Applications in Architecture, *Proceedings of CAAD Future09 Conference*, "Joining languages, cultures and visions", Montreal, Canada, June 17-19..
- Narahara T. (2009c). Bottom-up Design Inspired by Evolutionary Dynamics, *Proceedings of eCAADe 2009: (Education and Research in Computer Aided Architectural Design in Europe)*, *Computation: New Realm of Architectural Design*, Istanbul, Turkey, September 16-19.
- Narahara T. (2008). New Methodologies in Architectural Design inspired by Self-Organization *Proceedings of ACADIA (The Association for Computer-Aided Design in Architecture)*, *Silicon + Skin: Biological Processes and Computation*, Minneapolis (USA) October 2008..
- Narahara T. (2007a). *The Space Re-Actor: Walking a Synthetic Man through Architectural Space*, MS thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Narahara T. (2007b). Enactment Software: Spatial Designs Using Agent-based Models, *Proceedings of AGENT 2007: Conference on Complex Interaction and Social Emergence*. Sponsored by Argonne National Laboratory and Northwestern University, Northwestern University, Norris Center, Evanston, November 15-17.
- with Griffith, K., (2007c). Standardized Algorithms and Design Descriptions for "one-off" designs, *Proceedings of MCPC 2007: World Conference on Mass Customization & Personalization*. MIT Cambridge, Boston, October 7-9, 2007 at the Massachusetts Institute of Technology.
- Narahara T. (2007d). The Space Re-Actor: Walking a Synthetic Man through Architectural Space, *Proceedings of 25th Education and Research in Computer Aided Architectural Design in Europe (eCAADe) Conference*, Frankfurt, Germany, September 26-29.
- Narahara T. and K. Terzidis, (2006a). Multiple-constraint Genetic Algorithm in Housing Design, *Proceedings of the Association for Computer-Aided Design in Architecture (ACADIA) International Conference, Synthetic Landscapes Digital Exchange*, Louisville (USA) 12-15 October 2006, pp. 418-25.
- Narahara T. and K. Terzidis, (2006b). Optimal Distribution of Architecture Programs with Multiple-constraint Genetic Algorithm, *Proceedings of International Conference, SIGRADI 2006, Post Digital*, Santiago (Chile) 21-23 November 2006.