Chemotaxis-based Spatial Self-Organization Algorithms

A Thesis

Submitted to the Faculty

of

Drexel University

by

Linge Bai

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

August 2014

Dedications

I dedicate this thesis to my daughter, Genevieve Yi Lass. May you find science fascinating.

Acknowledgments

I would like to thank my advisor, Dr. David Breen, for his guidance and support during my eight year journey of graduate school, especially for shaping me into an independent scholar with an open mind. I truly enjoyed the intellectual freedom that he has provided for me. Additionally, I would like to thank Dr. Robert Gilmore, Dr. Santiago Ontañón Villar and Dr. Christian Kuehn for stimulating discussions and helpful ideas towards my research.

I would also like to thank my husband, Robert Lass, for his technical and financial support during my extended "childhood". Thank you for providing our daughter, Genevieve, with a comfortable environment to grow up in during these years. Many thanks to my parents for nurturing me into a curious person and for not interfering with my decisions.

Lastly, I would like to thank my friends and colleagues during my graduate school years. Many thanks to Manolya McCormick, Geoffery Oxholm, Kamelia Aryafar, Yuan Duan, Gabe Schwartz, Steve Lombardi, Yusuf Osmanlioglu, Ehsan Khosroshahi and many others, for having interesting conversations with me and for making my school years enjoyable.

I feel fortunate that I had the opportunity to attend graduate school and to obtain a PhD degree. To me, my graduate school experience is invaluable. It has turned me from a passive student who receives information into an active scholar who seeks knowledge. It gave me a computational mindset and the quantitative skills needed to evaluate situations and make decisions. I hope my eagerness for knowledge never fades and that my curious inner child never dies.

Table of Contents

List of Figures

List of Tables

Abstract

Chemotaxis-based Spatial Self-Organization Algorithms

Linge Bai

David E. Breen, Ph.D.

Self-organization is a process that increases the order of a system as a result of local interactions among low-level, simple components, without the guidance of an outside source. Spatial self-organization is a process in which shapes and structures emerge at a global level from collective movements of low level shape primitives. Spatial self-organization is a stochastic process, and the outcome of the aggregation cannot necessarily be guaranteed. Despite the inherent ambiguity, self-organizing complex systems arise everywhere in nature. Motivated by the ability of living cells to form specific shapes and structures, we develop two self-organizing systems towards the ultimate goal of directing the spatial self-organizing process. We first develop a self-sorting system composed of a mixture of cells. The system consistently produces a sorted structure. We then extend the sorting system to a general shape formation system. To do so, we introduce morphogenetic primitives (MP), defined as software agents, which enable self-organizing shape formation of user-defined structures through a chemotaxis paradigm.

One challenge that arises from the shape formation process is that the process may form two or more stable final configurations. In order to direct the self-organizing process, we find a way to characterize the macroscopic configuration of the MP swarm. We demonstrate that statistical moments of the primitives' locations can successfully capture the macroscopic structure of the aggregated shape. We do so by predicting the final configurations produced by our spatial self-organization system at an early stage in the process using features based

on the statistical moments. At the next stage, we focus on developing a technique to control the outcome of bifurcating aggregations. We identify thresholds of the moments and generate biased initial conditions whose statistical moments meet the thresholds. By starting simulations with biased, random initial configurations, we successfully control the aggregation for a number of swarms produced by the agent-based shape formation system. This thesis demonstrates that chemotaxis can be used as a paradigm to create an agent-based spatial self-organization system. Furthermore, statistical moments of the swarm can be used to robustly predict and control the outcomes of the aggregation process.

Chapter 1: Introduction

Self-organization is a process in which patterns at the macroscopic level emerge from numerous interactions at the microscopic level based on local interactions. The execution of the local interaction rules are based only on local information, without reference to the global pattern or guidance by an outside source [52]. Bird flocking and fish schooling are examples of self-organization phenomena, as in Figure 1.1. The coordinated behavior of a flock of birds or a school of fish emerges from low-level local interaction of the individuals, without a central leader in the system [20].

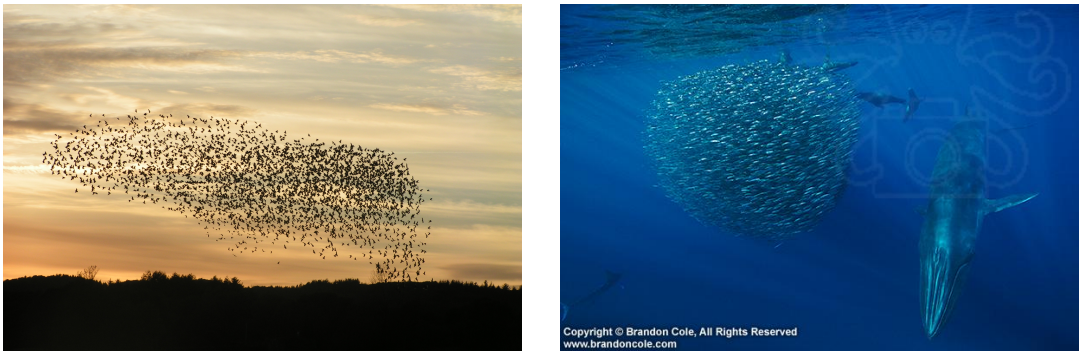


Figure 1.1: Self-organization in nature. (Photos by Christoffer Rasmussen and Brandon Cole)

Biological systems are based on the collective behaviors of a large number of simple and unreliable parts. In cell biology, each cell has a life cycle and is able to sense the environment, emit chemicals into the environment, proliferate and die. Cells interact with each other through the environment they live in. These behaviors at a microscopic cell level lead to the growth of macroscopic organisms. Cells proliferate, move and aggregate to create tissues and structures. This process, called morphogenesis, is one of the fundamental components

involved in the development of complex organisms [49], as in Figure 1.2. Chemotaxis is one of the mechanisms of cell interaction, with each cell emitting chemicals which diffuse into the surrounding environment. Neighboring cells detect the overall chemical concentration at their surfaces and respond to the chemical stimulus by moving either towards or away from the source [36].

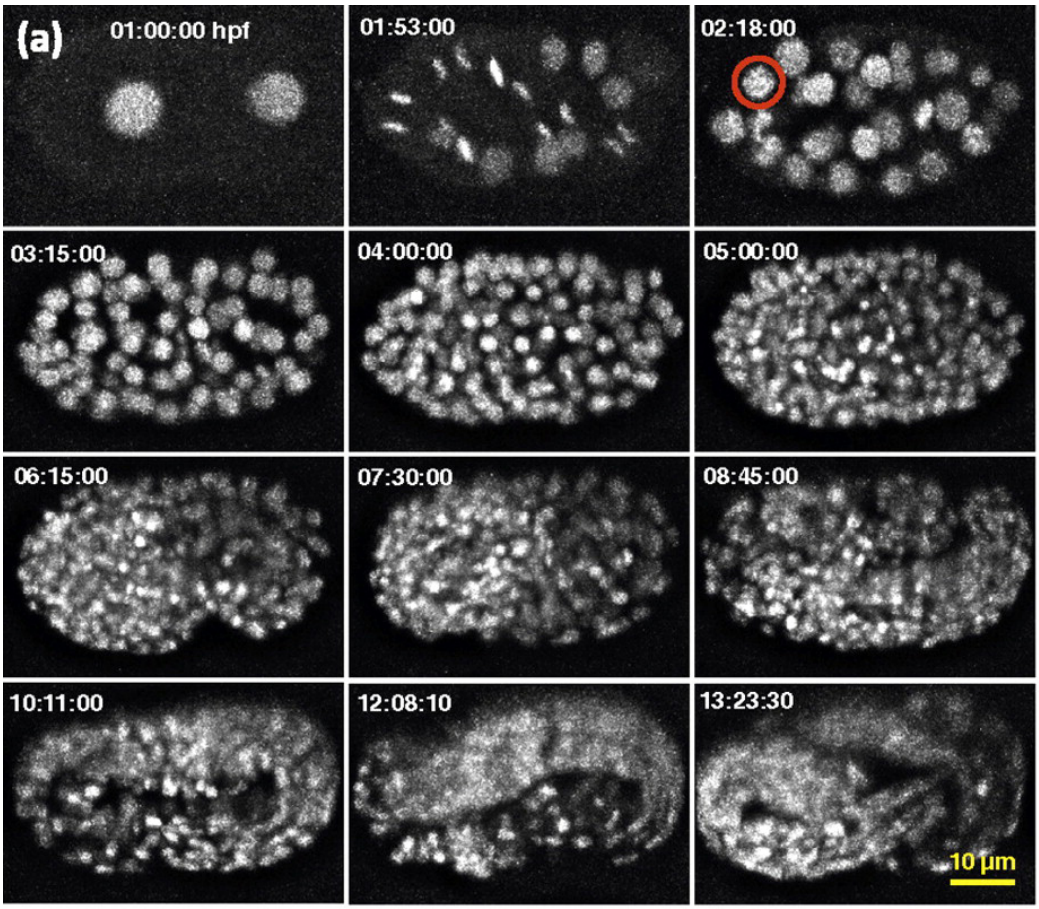


Figure 1.2: High-speed nuclear imaging of the nematode embryo. (Image by Yicong Wu)

In engineering, self-organization can provide a framework for distributed multi-agent systems, where a global goal is achieved by the coordination of lower-level agents. Compared with traditional top-down engineering approaches, a global predefined goal emerges from the simple local interactions of the agents. There is no central designer that distributes tasks

among each individual agent. The system is scalable and can adapt and/or recover from a sudden increase or decrease in the number of agents. Amorphous computing [1], spatial computing [13], swarm intelligence [15] and cellular automata [110] consist of biologically-inspired, novel engineering approaches and mathematical models for implementing self-organizing models. Morphogenesis modeling can be used to assist developmental biology and tissue engineering [63, 75]. Evolutionary computing [35], based on Darwinian evolution, has demonstrated novel approaches to solving problems where little is known about the solution space. Furthermore, algorithms based on morphogenesis and self-organization can be used in multi-robot control, reconfigurable robots and swarm robotics [79].

The main challenge when designing self-organization algorithms is to convey a global desired behavior to a large number of lower-level individual primitives. More specifically, given a task expressed at a global level, how can one design local interaction rules for lower-level components such that the local interactions among these components lead to the emergence of the global predefined behavior or structure? Current self-organizing shape formation algorithms aim to link local interactions at the microscopic level with the global desired behavior at the macroscopic level [16, 72, 73, 100]. However, existing algorithms either require some special global initialization of the environment and/or require that local primitives have a blueprint of the final shape to be formed. Moreover, existing self-organizing systems, instead of starting with desired global outcomes, begin by experimenting with different local interaction rules and then inspect the outcomes for desirable results. There have been difficulties in completely eliminating global information from the lower level primitives and 'reverse engineering' the local interaction rules that lead to specified outcomes. Truly local algorithms will provide the benefit of scalability and parallelizability to engineered systems. By eliminating central leaders, we also eliminate central points

of failure. This thesis provides the methodology of discovering local interaction rules for homogeneous autonomous agents that generate predefined, macroscopic shapes.

The first step of the methodology involved is developing a chemotaxis-based cell aggregation system that accurately simulates the behavior of actual, living cells. This simulator, which was developed by Dr. Manolya McCormick [40], produces cell aggregations of random form and structure. The aggregation system was extended to include two types of cells/agents. These two agents emit different types of chemicals into the environment and follow the chemical gradients sensed on their surfaces. Given proper parameter values, these agents self-organize from a random distribution into a central core of one type of agents surrounded by a layer of the other type of agents. Such sorting is observed during embryonic tissue formation and the differential clustering of cancer cells with different degrees of metastatic potential. We have developed a metric that quantifies the quality of the sorted structure. We performed numerous simulations with our algorithm in order to determine which parameters and associated values would produce sorting patterns with a quality above a certain threshold. We identified ranges for these parameters that guarantee the quality of the final sorted results.

While our self-sorting system can consistently produce a layered two-cell-type structure, it is only designed to produce one specific structure. This is achieved by changing the parameters of a fixed interaction. We aim to generalize this approach to spatial self-organization in order to robustly produce various user-defined shapes and structures. To achieve this goal, we propose a new approach to shape formation based on self-organizing shape primitives [5, 7], interacting via a chemotaxis paradigm.

In this approach, shapes are composed from a collection of lower-level self-organizing primitives. Macroscopic shapes are formed automatically by the aggregation of these simple

primitives. A shape primitive is represented by a small disc existing in a $2D$ environment. All shape primitives are initialized within a circle in the center of an arena with a uniform spatial distribution. Each primitive emits a finite "chemical" field, which accumulates in the environment. Each primitive detects the cumulative field at the eight receptors on its surface, and calculates the field gradient from these inputs. The gradient is used to determine the velocity of the primitive. Primitives move in the direction of the field's gradient with a speed proportional to the gradient's magnitude. A large-scale, user-defined shape then emerges from the combined actions of the individual primitives. The artificial chemical fields of individual primitives are explicitly defined as mathematical functions for each user-specified shape. Given the difficulty of designing the functions directly, genetic programming is used to evolve the local potential field functions for each shape, producing an automated shape formation capability.

Several principles are followed when developing the shape primitives. 1) Shape primitives are autonomous 'agents'. Each primitive is an independent entity that senses the environment, responds to it, and then modifies the environment and its internal state. There is no 'master designer' directing the actions/motions of the primitives. 2) Actions are based on local information. Each primitive emits a finite field that can be sensed only by other primitives within a certain range. The only information received by a shape primitive is gathered at its surface, namely the concentration of the cumulative field and contact with immediate neighboring primitives. 3) Shape primitives respond to information with prescribed behaviors. The actions performed by each primitive are the same, but the specifics of the individual actions are based on information received from the environment. 4) Shape primitives have no representation of the final, macroscopic shape to be produced. Primitives do not use information about the final shape to determine what actions to take.

Their actions are predetermined by the shape to be produced, but primitives do not carry or access information about the shape. 5) Primitives do not know their current global position nor their final position relative to the specified shape. 6) The macroscopic shape emerges from the aggregation of local interactions and behaviors. Rather than follow a plan to produce the shape, primitives sense and respond to the cumulative field concentration. This simple behavior, when combined with somewhat complex individual fields, will direct the primitives to take individual actions, based on local information, that ultimately direct them to aggregate into a user-defined, macroscopic shape.

We have evolved local field functions for a number of simple user-defined shapes, for example an ellipse, a diamond, an hourglass, the letter 'S', the letter 'b' and a triangle. Given the exploratory nature of our work, we have experienced failures when trying to make different target shapes. Moreover, a significant amount of computation is needed for the genetic process to produce the chemical field function for a particular shape. Additionally, during the aggregation process, each primitive starts with a random position. Given a specific field function and different initial conditions, self-organizing shape primitives do not always aggregate into the desired form. This system does not always consistently and robustly produce a desired shape. We have observed that there are instances when the primitives do not spatially self-organize into a unique shape, but instead form two or more stable final configurations.

In order to eliminate this bifurcation behavior and control the self-organizing shape formation process so that it consistently produces a single, desired shape, we analyze whole swarm populations at the global level in search of macroscopic, distinguishing attributes. This analysis identifies features, based on statistical moments of the agents' positions, that have significantly different values for different outcomes of a swarm aggregation. We dis-

covered that these statistical moments can be used to accurately predict the outcome of the self-organization process at an early stage of the shape aggregation [9]. Through our study of the dynamics of a swarm's statistical moments during the aggregation process, we have noted the connection between initial conditions and the final shape configuration of the swarm. We then discovered that biased, random initial conditions which meet specified constraints, i.e. have well-defined statistical properties, robustly yield simulations with a unique final outcome. Therefore, we are able to control the spatial self-organization process by specifying the statistical properties of its initial conditions.

This thesis contains descriptions of two self-organizing systems that we have developed on the way to the ultimate goal of completely controlling spatial self-organization. Our first attempt to achieve this goal developed a robust self-sorting system, inspired by cell sorting. Our second attempt developed morphogenetic primitives, which self-organize into user-specified macroscopic shapes based on local field functions. To achieve the final goal of directing spatial self-organization, we first identified macroscopic, distinguishing features by analyzing whole swarm populations at a global level. We compute the features based on the positions of the swarm's agents and accurately predict the final outcome of its aggregation [9]. Finally, we use these features (statistical moments of the positions) as constraints to create biased, random initial conditions, which leads to an approach for robustly directing spatial self-organization.

The contributions of our work will extend several fields and are the following:

- Self-sorting of heterotypic agents. We developed a self-organizing algorithm that guarantees agent sorting. Modeling and studying cell sorting is important to developmental biology and cancer research. It also provides paradigms for biology-based algorithms of self-organization behavior.

- Morphogenetic primitives. Rather than utilizing traditional shape modeling methods (e.g. parametric shape modeling, spring-mass deformable models, solid shape modeling), we developed an 'organic' method for modeling predefined macroscopic shapes with self-organizing primitives. This approach can be extended for and applied to the control of robotic swarms, motion specification of animated crowds and generative model creation.
- Investigation of statistical moments. We studied properties of the positions of all primitives as a whole. From this we discovered macroscopic, distinguishing features that may be used to characterize different final swarm configurations. Furthermore, from these features we are be able to confidently predict the final outcome of the system at an early stage during the shape formation process.
- Directing self-organizing shape formation with biased initial conditions. The results of the previous work lead to a control methodology that can be applied to self-organization of multi-agent shape formation, swarm robotics and sensor networks.
- Overall, this thesis describes a methodology for programming self-organizing primitives that robustly form into user-defined macroscopic shapes.

Chapter 2: Self-Sorting Primitives

2.1 Introduction

Our first effort to develop a robust self-organizing system produced a cell sorting simulation. The cell sorting simulation is based on a previously developed chemotaxis-based cell aggregation model (Figure 2.1), that successfully captures the cell behaviors that play important roles in $2D$ cell aggregation [40]. Figure 2.2 presents a comparison between an in vitro cell aggregation experiment and in silico aggregation results produced by out system. It can be seen that the cell aggregation model generates accurate but random cell aggregates similar to those produced by actual living cells.

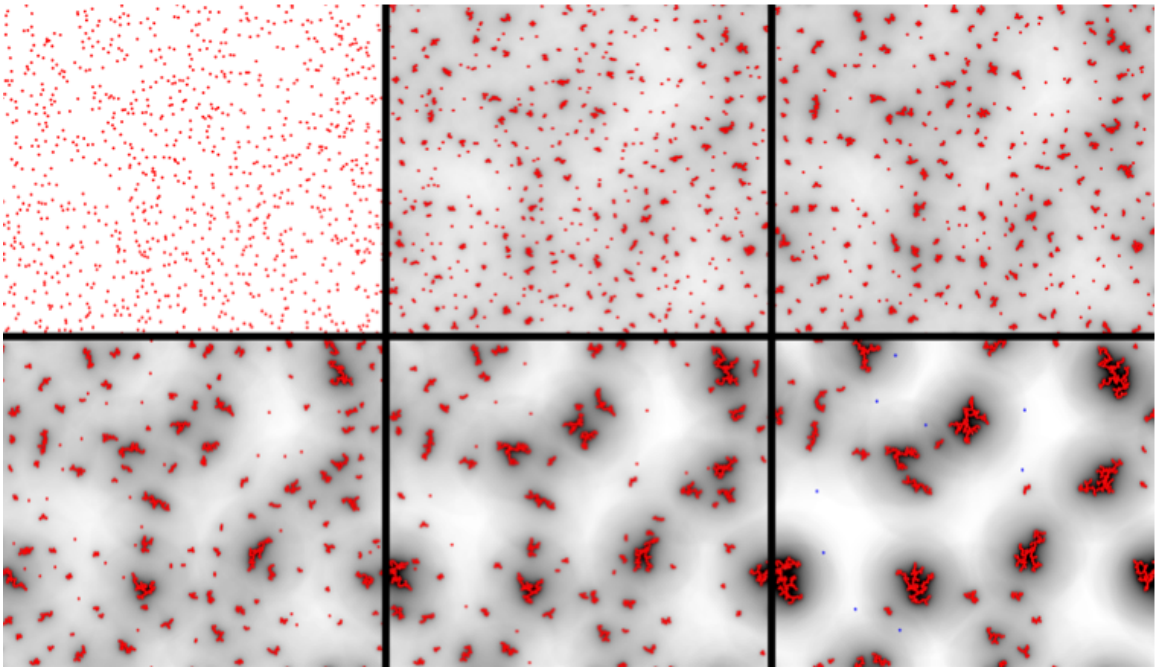


Figure 2.1: Chemotaxis-based cell aggregation simulation.

In the cell aggregation model, a cell's life cycle and behaviors are implemented as set of actions that are performed during each time step of a simulation. A single aggregation

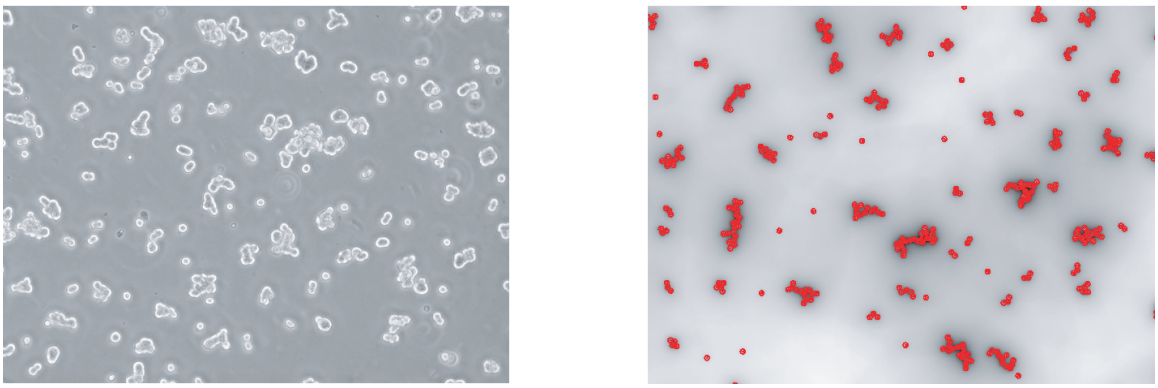


Figure 2.2: Comparison of live and simulated data. (left) Microscope image from *in vitro* experiment. (right) Visualization of simulated aggregation.

simulation is comprised of a series of these time steps. A cell's actions for each one of these time steps are outlined in Figure 2.3 and Table 2.1. Newly divided cells remain in a quiescent state for a user-defined period. These cells are not *Active* and cannot divide until they once again become *Active*. Immediately after cell division one of the new daughter cells separates from its sister cell. It moves with a user-defined velocity along the division axis for a user-defined period of time. Upon completion of this separation period the cell begins to emit a chemoattractant chemical and to respond to the chemoattractant gradient in its local environment. During the *Quiescent* phase there is some randomness in the cell's movement. These random motions vanish as the cell becomes *Active*. Based on the chemical fields produced by nearby cells a gradient is calculated and the cell/ aggregate moves in the gradient direction in response to it. A cell is capable of attaching to other cells upon collision. A cell's age is incremented at each time step. If the cell is apoptotic, a check is then performed to determine if the cell should die. If the state of the complete environment is to be saved for visualization purposes, the cell emits and stores its chemical field in the chemoattractant array. If in the *Active* stage, the cell determines if it is time to divide. If it

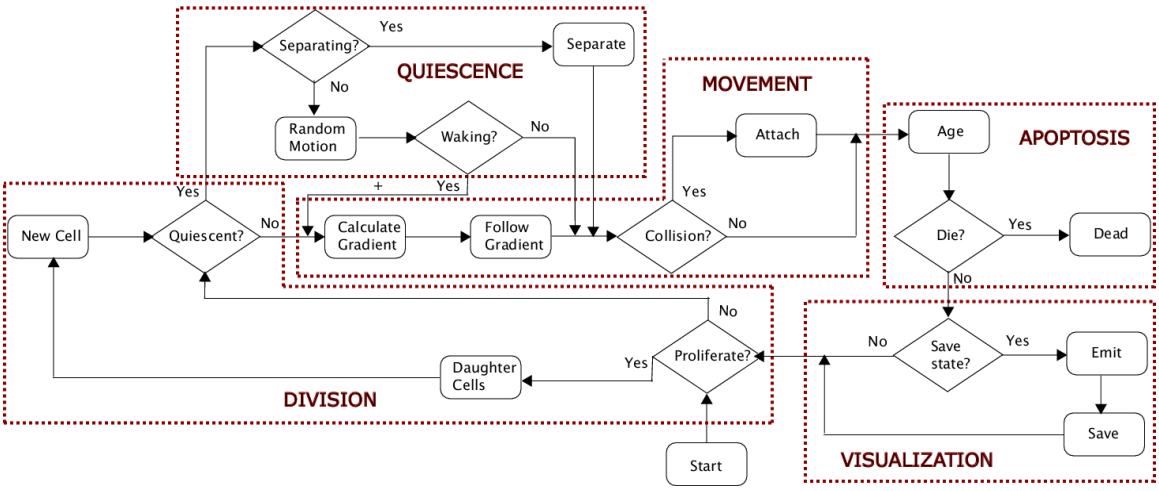


Figure 2.3: Computational flow of cell aggregation simulation per time step per cell.

Table 2.1: Actions taken by a virtual cell.

Function	Description
Emit()	Emit chemoattractants from surface
Sense(Coord x,Coord y)	Sense chemoattractant concentration at surface sensors
CalculateGradient()	Calculate the chemical gradient using the values read at 8 surface receptors
FollowGradient()	Move in the direction of the chemoattractant gradient
RandomMotion()	Take a random number of steps in a randomly chosen direction
Attach(Cell c1,Cell c2)	Attach to another cell after a collision
Proliferate(Cell c)	Divide and create two new cells
Age	Increment internal clock
Die(Cell c)	Shut down receptors and stop contributing/responding.

divides, its and its daughter cell's ages are set to zero, and they enter the *Quiescent* stage.

We extended the cell aggregation simulation system to include a chemotaxis-based heterotypic cell sorting model. In this model, two types of agents (T_1 and T_2) are first randomly placed in a toroidal environment. Agents emit different types of chemicals into the environment and follow the chemical gradients sensed on their surfaces. Given the proper parameters, these agents self-organize from a random distribution, in Figure 2.4 (left), into a central core of T_1 (blue) agents surrounded by a layer of T_2 (red) agents, as in Figure 2.4 (right). We aimed to analyze and identify important parameters in the cell sorting process

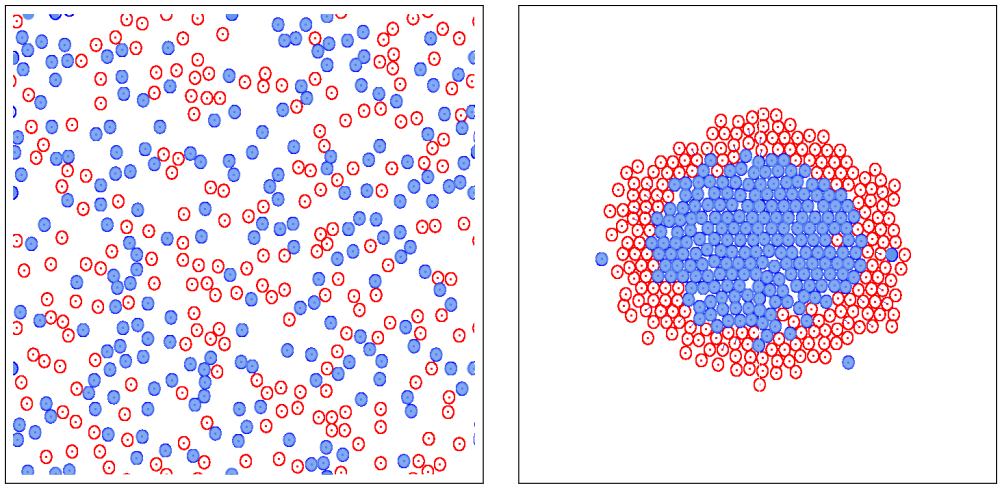


Figure 2.4: Initial and ending configurations of a self-organizing two-agent-type system (blue T_1 agents and red T_2 agents). The quality value of the initial state is 0.52 and final structure is 1.60.

that robustly produce the desired sorted structure.

The dynamic sorting of heterotypic populations, which results in the enclosure of one cell grouping by another, leads to the organization of tissues during morphogenetic processes such as embryogenesis, organogenesis and tumorigenesis [49]. For example, such sorting/organization is observed during the differential clustering of cancer cells with varying degrees of metastatic potential. Beyond making contributions to developmental biology, cancer research and biomedical engineering, the simulation of cell sorting can also provide paradigms that may be used to design biology-based algorithms for self-organizing behavior, such as directing the movements of a robot swarm. It could be used in a multi-robot rescue or surveillance system, where one group of robots needs to surround another group of robots. Additionally the algorithm could be employed to direct a sensor network to autonomously configure itself into a specific spatial pattern. In the future, the agent-sorting algorithm could be utilized to form a $3D$ wire-like structure with reconfigurable microbots, by noting that the $2D$ sorted structure resembles the cross-section of a wire. Having microbots sort

themselves perpendicular to and along a path-line would produce a structure with a central filament surrounded by a casing.

2.2 Related Work

A number of computational models for simulating biological cell sorting in 2D or 3D have been developed. A majority of the research uses the Cellular Potts Model (CPM), a lattice-based model based on the large-Q Potts model, to investigate biological cell sorting in both 2D [51] and 3D [71, 50]. These models assume that differential adhesion, i.e. the Differential Adhesion Hypothesis (DAH) [93], is the main cause of sorting in heterogeneous cell mixtures. The CPM model has a global energy function, based on contact energies between different cell types and the extra-cellular matrix, that when minimized can sort two kinds of cells. In similar work, Hogeweg [54, 55] studied the relationship between cell differentiation and cell adhesion, and its affect on modes of morphogenesis within a cellular automata framework. More recently cell sorting models based on self-propelled particles [25, 102] have been developed. These models have been used to study the influence of intrinsic cell motility [14] and the relationship between intercellular adhesion and surface tension [103].

Based on DAH and control theory, Kumar et al. used an artificial differential potential to direct the segregation of heterogeneous agents into two separated groups [64]. Agents move towards other agents of the same type and away from agents of a different type. Other research in multi-robot formation control or pattern generation usually involves a lead-following approach [26], a global potential field [57] or requires a GPS system [11].

Computational biology models have also been used for self-organizing geometry and evolutionary computing. Fleischer explored a cell-based developmental model for self-organizing geometric structures [44, 45]. Eggenberger-Hotz [33, 56] proposed the use of

genetic regulatory networks coupled with developmental processes for use in artificial evolution and was able to evolve simple shapes. The combination of artificial evolutionary techniques and developmental processes provides a comprehensive framework for the analysis of evolutionary shape creation. Nagpal et al. [72, 73] present techniques to achieve programmable self-assembly. Cells are identically-programmed units which are randomly distributed and communicate with each other within a local area. In this approach, global-to-local compilation is used to generate the program executed by each cell, which has specialized initial parameters. This work has been extended and applied to collective construction based on a swarm of autonomous robots and extended stigmergy [108]. Work has recently begun on bio-inspired self-organizing heterogeneous systems. Doursat [28] utilizes artificial system growth inspired by embryogenesis as a model for evolutionary design. Beal [12] presents the concept of Functional Blueprints, an engineering approach for specifying the desired performance of and the means of incrementally correcting deficiencies in grown systems.

A computational model of chemotaxis-based cell aggregation [38, 40] provided a framework, paradigm and interaction constructs for designing the correct set of operations and parameter values that produce the desired sorted result in our research. In this model, a virtual cell is designed as an independent, discrete unit with a set of physiologically relevant parameters and actions. Each cell is defined by its size, location, rates of chemoattractant emission and response, age, life cycle stage, quiescent period, proliferation rate and number of attached cells. All cells are capable of emitting and sensing chemoattractant chemicals, moving, attaching to other cells, dividing, aging and dying.

Our agent sorting algorithm stands apart from previous work in many ways. While Nagpal et al.'s [72, 73] work is bio-inspired, it focuses on agents that fold and grow into

specific shapes and require specialized markers in the environment. Werfel and Nagpal's extended stigmergy approach [108] utilizes *homotypic* agents that access a global coordinate system, as well as a shared shape description, to produce objects with a pre-defined configuration. Other robotic swarm techniques [11, 26, 57] utilize global or centralized information, again with *homotypic* agents. Our work focuses on sorting a population of *heterotypic* agents [8]. The Kumar et al. technique [64] is able to de-mix a two-type group of agents into two separate amorphous clumps. This work does not address the problem of directing a heterotypic swarm into a specific tightly-packed, sorted formation. The work on more complex systems that form into shapes [12, 28, 33, 44, 45, 56] employs multiple mechanisms of morphogenesis, most importantly proliferation and differentiation, i.e. the agents in these systems reproduce and change type. These two features preclude these kinds of algorithms for use in our target application, robot swarm formation, since robots cannot reproduce. The desire to keep the actions of the individual robots as simple as possible also rules out differentiation as an agent attribute.

In comparison to previous work, our approach is unique in that it creates a tightly-packed sorted structure from an initially mixed population. It is able to achieve this goal purely through local interactions and simple behaviors. All agents of the same type are homogeneous. There is no leader agent, and all agents of the same type are treated equally during the computation. The agents in our algorithm do not know their position in the world, nor do they have a representation of the final macroscopic shape. No global information or environmental markers are required to produce the desired result. Our algorithm is an extension of a chemotaxis-based cell aggregation model; we employ chemotaxis as a paradigm for controlling heterotypic agents. Agents simply emit chemicals into their environment, follow the cumulative chemical gradients that they sense at their surfaces, attach,

detach, and self-organize into a sorted, layered structure.

2.3 Chemotaxis-Based Agent Sorting Model

We have chosen to focus on chemotaxis, cell motility and DAH [93] as the main mechanisms to include in an algorithm that produces sorting behavior in self-organizing heterotypic agents. In our scenario, there are 200 T_1 (blue) agents and 200 (red) T_2 agents in a toroidal environment, i.e. the top edge of the computational arena is connected with the bottom edge and the right edge is connected to the left edge, which is 500×500 units in size. To be consistent with cell dimensions, 1 unit is equivalent to 1 μm. All agents are the same size (radius = 6 units, a size consistent with many cells [2]).

Both T_1 and T_2 agents age (i.e. maintain an internal clock) during the simulation, and their total numbers stay fixed. T_1 agents emit two chemicals (C_1 and C_2) into the environment, but only respond to chemical C_1. T_2 agents do not emit any chemicals, and only respond to chemical C_2. Collisions between agents may form an attachment, and once the attachment is formed, all agents in the aggregate move with the same velocity.

A single simulation is comprised of a series of time steps, with each step equaling 60 seconds, and runs for a simulated 90 hour period. Important parameters in the model are λ_i (magnitude of response to chemoattractant i), P_R (probability of responding to a chemoattractant), T_S (time between a T_1 agent's first attachment and production of chemoattractant C_2), P_{Attach} (probability of attachment) and P_{Detach} (probability of detachment).

Our agent-sorting algorithm has been implemented by modifying a previously developed chemotaxis-based cell aggregation simulation system [38, 40]. The work here extends the system by defining multiple cell types with more complex behaviors. In the system, chemoattractants are secreted from the agent's surface symmetrically and diffuse radially. The concentration of the chemoattractant initially secreted by a single agent at its surface

is N_0 *molecules/units*^{2} ($N_0 = 270$ for both chemoattractants, i.e. C_1 and C_2, in our experiments [80]). We assume that a constant chemical concentration is maintained at the agent's surface, creating a static, circular chemical concentration field around each agent. Given this assumption, the chemoattractant concentration C_i within the field drops off as $1/r$, where r is the distance from the agent surface [23]:

$$C_i(r) = \frac{N_0}{1+r}, \quad i = 1, 2. \tag{2.1}$$

It is known that once the chemoattractant concentration falls below a certain value, cells will no longer respond to the chemoattractant [86]. This phenomenon allows us to define a *finite* field around an agent with a radius of R_{Max} (300 units for our simulations). Any agent within a distance less than R_{Max} to another agent is influenced by the chemoattractant emitted by the other agent. An agent that is further away than R_{Max} from an emitting agent does not detect its chemoattractant and the detecting agent's motion is not affected by the emitting agent.

2.3.1 Chemotaxis

Chemoattractant C_1 is produced by all T_1 agents at all times. A T_1 agent's production of chemoattractant C_2 begins after a certain amount of time (T_S, 18 hours in our simulations) after it has attached to another agent. C_2 emission from T_1 agents then steadily increases until it reaches a maximum rate (C_0) at 24 hours. T_1 agents are attracted to chemical C_1 and T_2 agents are attracted to chemical C_2. In this sequence of events, T_1 agents first attract each other by emitting and responding to chemoattractant C_1. This allows them to come together to form a single "blue" aggregate of T_1 agents. As they begin to attach to each other, T_1 agents then begin to emit chemoattractant C_2 (18 hours after their

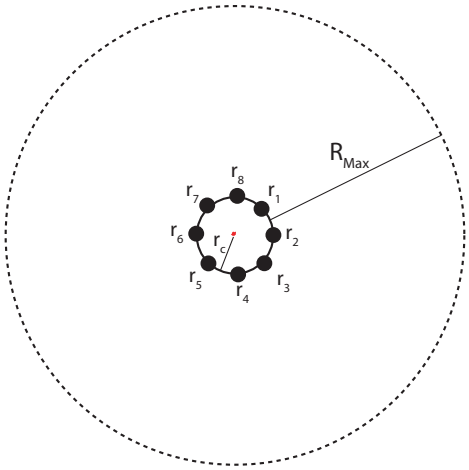


Figure 2.5: A single agent has radius r_c and eight chemoattractant receptors identified as r_i. The agent's chemical field cannot be sensed past R_{Max}.

first attachment); thus then attracting T_2 agents, which form around and attach to the T_1 aggregate, as seen in Figure 2.9.

Agents emit 'chemicals' into the environment. The 'chemicals' conceptually diffuse out from the agent's surface into the surrounding environment within a certain distance of influence in all directions in 2D. In our system the field function is truncated at a fixed distance ($R_{max} = 200$ units) in order to keep the agent interaction finite and local.

Each agent has eight receptors evenly distributed on its surface. The placement of the receptors, within the agent's local coordinate system, begins at 45° in the upper right and proceeds clockwise. The overall chemical gradient sensed by an agent is calculated with the

following equations,

$$C_x = \frac{\Lambda_2 - \Lambda_6 + \frac{\Lambda_1 - \Lambda_5 - \Lambda_7 + \Lambda_3}{\sqrt{2}}}{2 * r_c}$$
$$C_y = \frac{\Lambda_8 - \Lambda_4 + \frac{\Lambda_7 + \Lambda_1 - \Lambda_5 - \Lambda_3}{\sqrt{2}}}{2 * r_c}$$

$$\nabla C = (C_x, C_y) \tag{2.2}$$

where Λ_i is the chemical concentration sensed at receptor r_i and r_c is the agent's radius (5 units for our examples). The locations of the numbered receptors are provided in Figure 2.5.

Using a biology-based assumption that cells quickly reach a terminal velocity because of the viscous drag imposed by their environment, the velocity of a chemotactically stimulated agent is directly proportional to the chemical gradient of the cumulative chemical field (∇C_i^{cum}) detected at the agent's surface. The chemical concentration of the cumulative field at any point in space $\mathbf{X}$ is simply the sum of the individual chemical fields emitted by the agents that are within a distance R_{Max} of $\mathbf{X}$. An agent's velocity $\mathbf{V_i}$ is calculated as

$$\mathbf{V_i} = \lambda_i * \nabla C_i^{cum}. \tag{2.3}$$

The velocity $\mathbf{V_i}$ of an individual agent is clamped to 1 unit/minute, a maximum velocity consistent with a typical chemotactic cell response if the units are considered to be microns [41]. The magnitude of an agent's response to a chemoattractant is defined with parameters λ_1 (in response to C_1) for T_1 agents and λ_2 (in response to C_2) for T_2 agents. A stronger response, i.e. a greater value of λ_i, makes agents move faster and leads to shorter

aggregation times. When the chemotactic interaction between agents is weaker (i.e. for low values of λ_i), slower aggregation behavior is observed. Given the velocity calculated by Equation 2.3, at each time step Δt of a simulation a displacement $\mathbf{\Delta x_i}$ is calculated for each type of agent i,

$$\mathbf{\Delta x_i} = \mathbf{V_i} * \Delta t. \tag{2.4}$$

Different response rates (λ_1 and λ_2) are assigned to each type of agent for each chemoattractant. The difference in the response rates of T_1 and T_2 agents to C_1 and C_2, as described in [76], is an important feature of our agent-sorting algorithm. We defined λ_1 to be larger for T_1 agents (10.0) than λ_2 for T_2 agents (1.0) to induce T_1 agents to aggregate faster and form the core of the aggregate. λ_2 for T_1 agents is 0 and λ_1 for T_2 agents is 0, so these agents do not respond to these chemoattractants.

At each time step we probabilistically determine if the agent should respond to the gradient, based on probability P_R. If it is determined that agents should not respond to a gradient, or if no chemical gradient is present, the agent takes a random step of 1 to 6 units. This feature implements a type of biased random walk that is influenced by the strength of the agents' chemotactic response [58]. For T_1 agents, this probability is constant at 50%.

We found through experimentation that P_R for T_2 agents needed to be a function of the chemoattractant concentration sensed at the agent's surface in order to consistently produce the desired final result. As the concentration of C_2 increases, so does the probability that a T_2 agent will move in the direction of C_2's gradient. We use the function

$$F(C_2^{avg}) = \frac{1}{2} - \frac{1}{2} * cos(\pi * min(C_2^{avg}, C_{max})/C_{max}), \tag{2.5}$$

to define P_R for T_2 agents, where C_{max} is 270 $molecules/units^2$ in our simulations. C_2^{avg}

is the average C_2 concentration sensed at the eight receptors on a T_2 agent's surface. The equation produces a smoothly increasing, then clamped, probability using a cosine function that begins at zero and increases to 1 at C_{max}, and remains 1 above this concentration value, as in Figure 2.6.

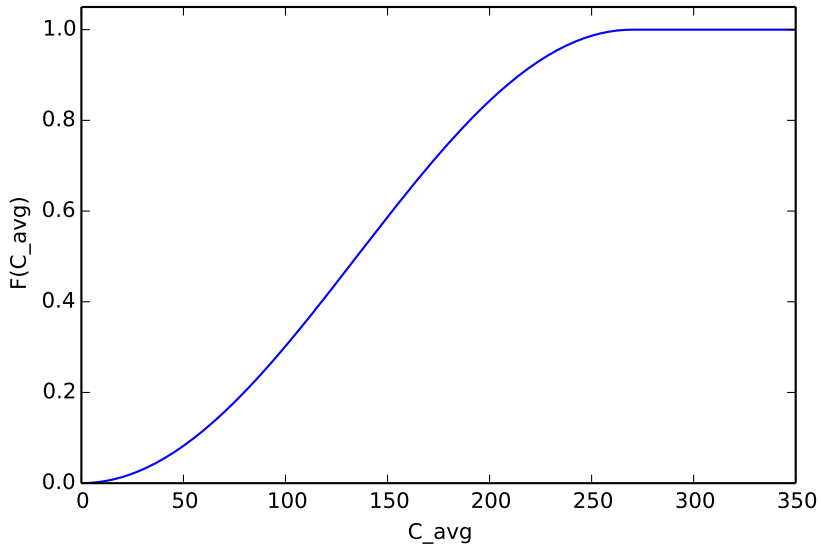


Figure 2.6: Probability of T_2 agents responding to a chemoattractant.

Agents may attach to each other and form aggregates of increasing size. Once an aggregate is formed, all of the agents within the aggregate move with the same velocity $\mathbf{V_{aggr}}$, as defined by

$$\mathbf{V_{aggr}} = \mathbf{V_{avg}}/M, \tag{2.6}$$

where $\mathbf{V_{avg}}$ is the average of the velocities calculated for each agent in the aggregate as if it were unattached, and M is the mass of the aggregate, i.e. the number of agents in the aggregate.

2.3.2 Attachments and Detachments

We assume that, similar to newly formed cells [2], agents are initially quiescent and are unable to form attachments. Specifically, our agents do not form any attachments for the first 5 simulation hours. Additionally, we assume that agents do not form attachments unless they are in contact with at least four other agents. Otherwise their own kinetic energy is able to overcome the adhesion of just a few agents.

T_1 agents probabilistically start forming aggregates upon collision after 5 hours and this triggers the production process for another binding chemical. Five hours after a T_1 agent forms an attachment with another T_1 agent, it can start attaching to T_2 agents. These T_1-T_2 attachments trigger T_2 agents to also produce a binding chemical and in another 5 hours they are able to attach to other agents as well.

If an agent is capable of attaching to another agent (i.e. it is older than 5 hours and has the appropriate type and number of neighboring agents, 4 or more), it makes the attachment upon collision with probability P_{Attach}. This probability is a function of the type of agents involved in the collision; thus implementing a form of differential adhesion.

$$P_{Attach} = \begin{cases} 100\% & if\ both\ are\ T_1 \\ 50\% & if\ one\ is\ T_1\ and\ one\ is\ T_2 \\ 10\% & if\ both\ are\ T_2 \end{cases} \tag{2.7}$$

T_1 agents have a greater chance of forming attachments and therefore create bigger aggregates than T_2 agents. T_1 agents always attach with each other. T_1-T_2 attachments only are formed during half of the T_1-T_2 collisions. T_2 agents only attach with each other 1 out of 10 collisions. This behavior creates bigger and growing aggregates of T_1 surrounded by mostly

single agents of type T_2.

There is some probability that the outer layer agents of an aggregate can detach from the aggregate. We model this behavior in our simulation system with a probability of detachment, P_{Detach}. Agents with 3 or fewer neighbors are not considered fully surrounded and have a 30% probability of detaching from their neighbors. When an agent detaches it takes a random step of 1 to 6 units away from the aggregate to which it was previously attached. In the next time step the detached agent detects and responds to the chemoattractant gradients. Since the length of the random step is at most 1 agent radius, the separated agent usually returns and attaches to the same aggregate within a short amount of time. Since the agent follows the gradient after separation, detachments give the agent the ability to slide over their neighbors and attach to a different location on the aggregate. Detached agents will move in a direction towards a greater concentration of agents. We observed that including agent detachments in the algorithm led to more circular final aggregates.

2.3.3 Agent Actions

The actions, and their order, taken by each agent at each time step of a sorting simulation are detailed in Figures 2.7 and 2.8.

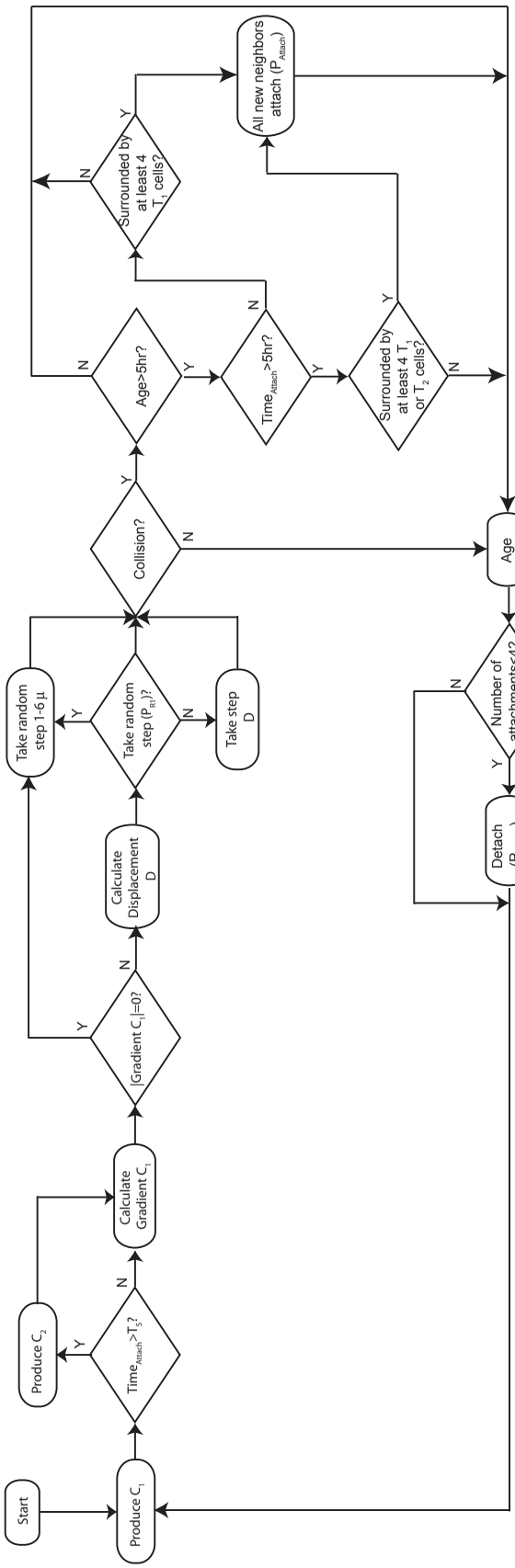


Figure 2.7: Computational flow chart for a T_1 agent during one time step of the agent sorting algorithm.

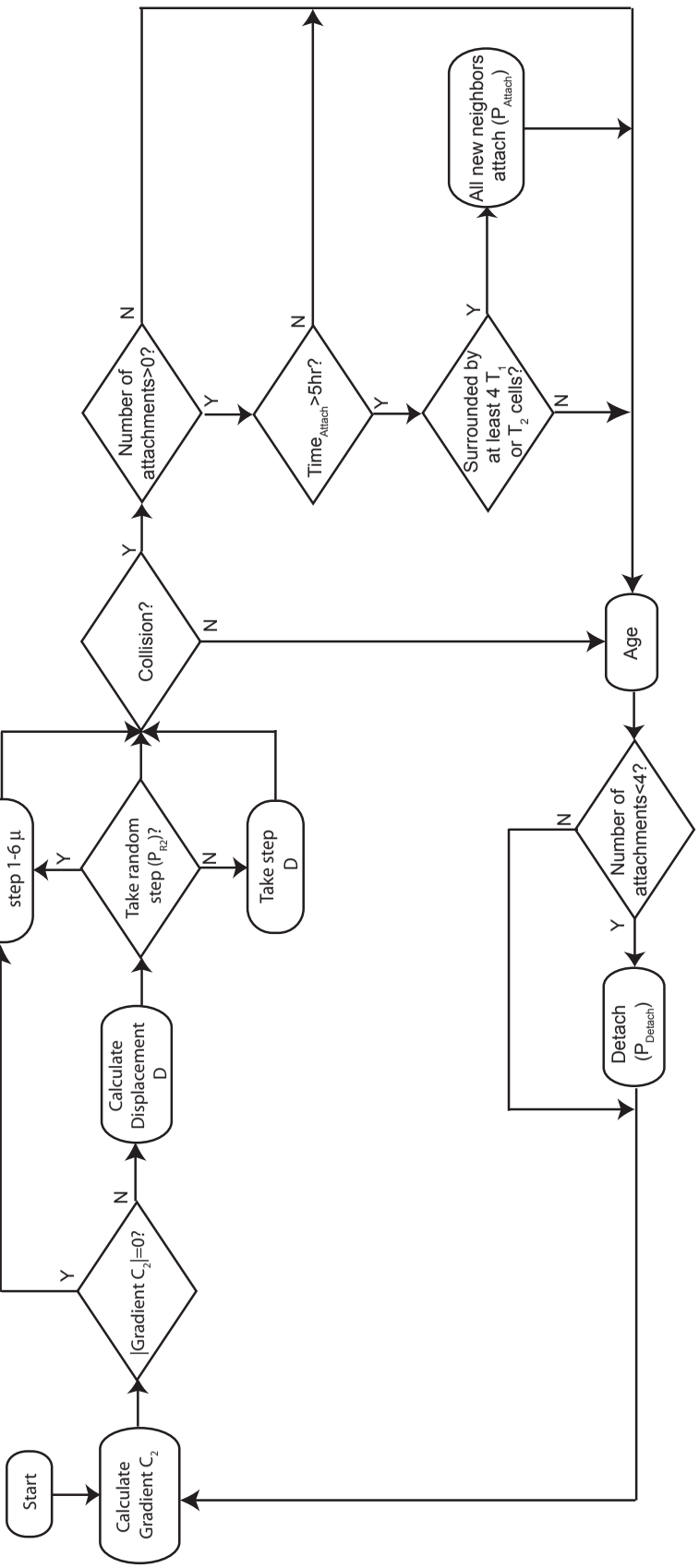


Figure 2.8: Computational flow chart for a T_2 agent during one time step of the agent sorting algorithm.

T_1 Agent

At the beginning of a simulation time step, a T_1 agent emits chemical C_1 into its environment. If the T_1 agent is attached to another agent and if the time since that attachment occurred is later than T_S (18 hrs), the agent also emits chemical C_2 into the environment. The chemical C_1 is sensed at a T_1 agent's surface and a concentration gradient is calculated. If no chemical gradient is sensed at its surface, the agent takes a random step. If a gradient is sensed, with the probability P_R (50%) the agent moves in the direction of the gradient with a velocity proportional to λ_1 (10.0) based on Equations 2.3 and 2.4, otherwise it takes a random step of 1 to 6 units.

If no collision occurs, the agent increments its age and goes to the detachment stage. If a collision has been detected and the T_1 agent's age is less than 5 hours, the agent does not form an attachment and goes to the age increment stage. Otherwise the time since the first attachment is checked. If this time is less than 5 hours the agent will form attachments only if it is surrounded by 4 or more T_1 agents, i.e. it only forms attachments with other T_1 agents. P_{Attach} is 100% for T_1-T_1 attachments. If the time since its first attachment is greater than 5 hours the T_1 agent will attach to both T_1 and T_2 agents, if it is surrounded by 4 or more of them. P_{Attach} is 50% for T_1-T_2 attachments.

Once the attachment stage is complete, the agent increments its age. If the agent has less than four attachments, with a 30% probability the agent will detach from its neighbors. Note that an agent may become attached to another agent via the other agent's attachment process, and therefore may have fewer than four attachments.

T_2 Agent

At the beginning of a simulation time step, a T_2 agent senses chemical C_2 at its surface and calculates the chemical's gradient. If no chemical gradient is present, the agent takes a random step. The gradient is scaled by λ_2 (1.0) and a displacement is calculated using Equations 2.3 and 2.4. With probability P_R the T_2 agent moves in the direction of the gradient of C_2, otherwise it takes a random step of 1 to 6 μms. P_R is an increasing function of the C_2 concentration sensed on the agent's surface, defined by Equation 2.5.

If no collision occurs, the agent increments its age and goes to the detachment stage. If a collision has been detected, but the T_2 agent is unattached, the agent goes to the age increment stage. In other words a T_2 agent does not attach itself to another agent until another agent attaches to it first. If it is attached, the time since the first attachment is checked. If this time is greater than 5 hours the agent will form attachments only if it is surrounded by 4 or more agents. Attachments are formed with T_2 agents with a 10% probability, and with T_1 agents with a 50% probability. Once the attachment stage is complete, the agent increments its age. If the agent has less than 4 attachments, with a 30% probability the agent will detach from its neighbors.

2.4 Results

We performed numerous simulations with our algorithm in order to determine which of its parameters and associated values would produce the sorting patterns presented in [47, 94]. We concluded that λ_i (magnitude of response to chemoattractant i), P_R (probability of responding to a chemoattractant), T_S (time between a T_1 agent's first attachment and production of chemoattractant C_2), P_{Attach} (probability of attachment) and P_{Detach} (probability of detachment) have the greatest impact on agent sorting outcomes. The parameter

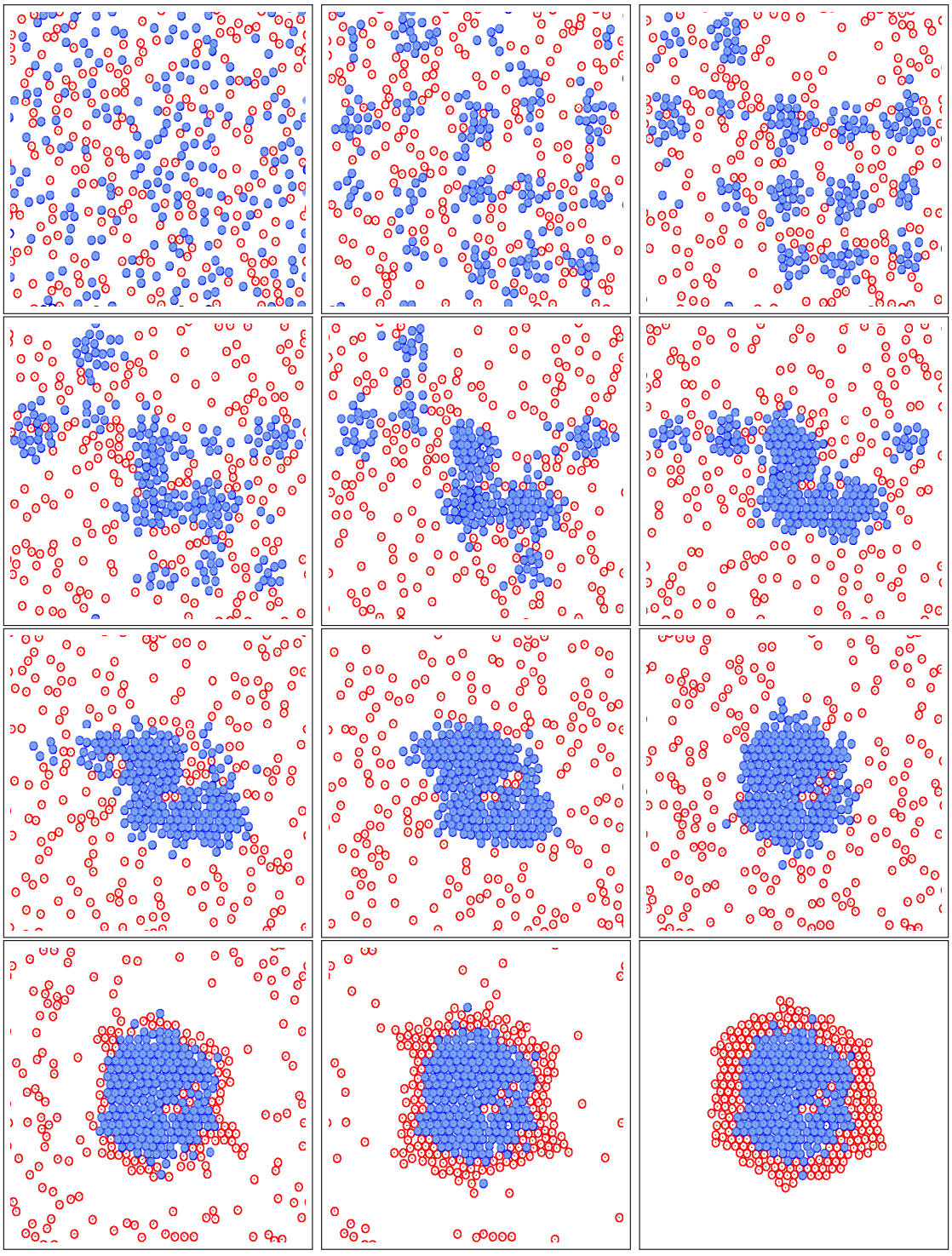


Figure 2.9: Simulation of self-organizing heterotypic agents sorted into a layered structure. T_1 (blue) agents emit a chemical that attracts other T_1 agents, forming a central blue core. Then they emit a second chemical that attracts T_2 (red) agents, forming the outer layer.

Table 2.2: Parameter values that produce sorted structure with the highest quality value.

Type	λ_1	λ_2	P_R	P_{Detach}	T_S	R_{Max}
T_1	10.0	0.0	0.5	0.3	18 hrs	300
T_2	0.0	1.0	$F(C_2)$	0.3	-	300

values that produce the desired result pictured in Figures 2.4 and 2.9 are listed in Table 2.2 and Equation 2.7. These values show that T_1 (blue) agents strongly respond to C_1 ($\lambda_1 = 10$), but do not respond at all to C_2 ($\lambda_2 = 0$). T_2 agents respond weakly to C_2 ($\lambda_2 = 1.0$), and do not respond to C_1 ($\lambda_1 = 0$). Since P_R is the probability that agents will respond to a chemoattractant, it can be seen that T_1 agent's response to C_1 remains constant at 50%. A T_2 agent's response to C_2 is an increasing function (Equation 2.5) of the C_2 concentration sensed at the agent's surface. 18 hours is the time needed between a T_1 agent's first attachment and production of chemoattractant C_2 (T_S) for the desired sorting to occur.

As seen in Figure 2.9, the two agent populations are initially mixed. Since the blue T_1 agents strongly and always attract each other, they quickly form small aggregates, which then ultimately come together to form a single blue grouping. Red T_2 agents initially move randomly in the environment before the production of C_2 begins. After T_S and the start of C_2 production, T_2 agents become more strongly attracted to T_1 agents. T_2 agents then close in to form a tightly packed layer around the T_1 agents. Note that the agents move on a hexagonal grid [38] with a 1 unit distance between each grid location. This low-level constraint influences the aggregate's resulting large-scale outer shape. The low-level structure of the hexagonal grid can be seen in the overall shape of the final sorted aggregate.

All of our sorting simulations required approximately 15 CPU-minutes of computation

time on an Apple MacBook with an Intel dual core 2.0 GHz processor and 1GB of RAM.

2.5 Parametric Studies

We categorize the parameters in our system into agent and environment parameters. Agent parameters include λ_i, P_{Attach}, P_{Detach}, T_S, P_R and R_{Max}, and are associated with each individual agent. Environment parameters include agent number, environment size, noise and initial conditions. Once the desired sorting result was produced, a series of parametric studies were performed to explore the influence of these parameters on the shape and structure of the final aggregates.

2.5.1 Sorting Quality Evaluation

In order to quantitatively evaluate the quality of the emergent layered sorted structure, we have defined an evaluation measure based on the distribution of the two types of agents in the final aggregate. This measure produces a scalar that is able to quantify the quality of the sorted structure and allows us to compare and rank different results.

Since the desired structure is a round disk with T_1 (blue) agents in the center and T_2 (red) agents surrounding the center, we have devised an evaluation measure that is maximized when the sorting process produces the sought after result. The evaluation measure is based on the location of the red and blue pixels in the final image of the aggregate. The first step in its computation involves calculating the centroid of the blue pixels, $Center$. The average distance between the centroid and the blue pixels, then the red pixels is calculated,

$$R_{avg}^{blue} = \frac{1}{n} \sum_{pixel_i^{blue} \in T_1} Dist(pixel_i^{blue}, Center) \tag{2.8}$$

$$R_{avg}^{red} = \frac{1}{m} \sum_{pixel_i^{red} \in T_2} Dist(pixel_i^{red}, Center), \tag{2.9}$$

where n is the number of blue pixels and m is the number of red pixels, and $Dist()$ is the Euclidean distance between a pixel and the centroid of the blue pixels. Given these values we calculate the standard deviation of the distances between the individual blue and red pixels to the centroid,

$$\sigma^{blue} = \sqrt{\frac{1}{n} \sum_{pixel_i^{blue} \in T_1} (R_i^{blue} - R_{avg}^{blue})^2} \tag{2.10}$$

$$\sigma^{red} = \sqrt{\frac{1}{m} \sum_{pixel_i^{red} \in T_2} (R_i^{red} - R_{avg}^{red})^2}, \tag{2.11}$$

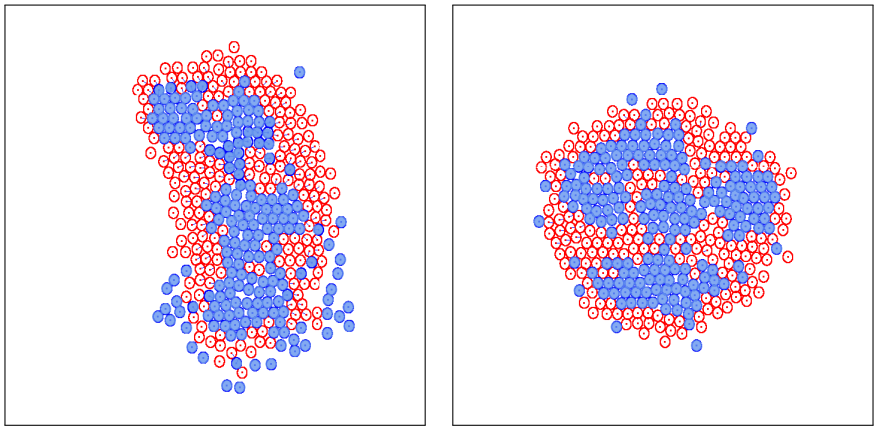
where R_i^* is the distance between pixel i and $Center$. Finally, given the standard deviations a quality measure is defined that has a maximum value when the desired sorted structure is generated,

$$quality = 100/(\sigma^{blue} + \sigma^{red}). \tag{2.12}$$

We should note that since the sorting is computed in a toroidal environment, an aggregate must first be centered in its image [3] (See Appendix A for the details of the centering algorithm.) before the quality measure is computed. What we deemed as acceptable sorted results can be seen in Figures 2.4 (right) (quality = 1.60), 2.9 (bottom-right) (quality = 1.55), 2.12 (right) (quality = 1.42) and 2.14 (right) (quality = 1.41). These results were considered "acceptable" via visual inspection, because it could be seen that well over 95% of the agents were divided into two contiguous groups. Note that all of these results have a quality value above 1.4. Based on this observation we empirically define 1.4 as the lower threshold value of the sorting quality metric for acceptable sorting results.

2.5.2 Agent Parameters

In order to explore the influence of the agent parameters, we initialize the agents with the optimal parameter values listed in Table 2.2, and then modify the value of a single parameter to demonstrate its effect on the sorting behavior.



(a) quality=0.76 **(b)** quality = 1.08

Figure 2.10: Effect of chemoattractant gradient response λ_1 and T_S on sorting. (a) $\lambda_1 = 1.0$; (b) $T_S = 6$ *hours*.

λ_i

Figure 2.10(a) presents the effect of chemoattractant gradient response parameter λ_1 on the sorting results. The figure shows disrupted aggregation when the response of T_1 agents to the C_1 chemoattractant chemical is reduced. Slower blue T_1 agents cannot form into a single well-sorted aggregate, before being engulfed by the red T_2 agents. Increasing λ_1 does not affect the sorting outcome, it just makes the blue agents form a single aggregate more quickly. There is a relationship between λ_1 and T_S. λ_1 determines how quickly blue agents form a single aggregate. They must form this aggregate before the red T_2 agents begin to move towards blue agents, an action whose timing is determined by T_S. Changing λ_2 does not significantly affect the sorting result. As long as the blue agents have enough time to

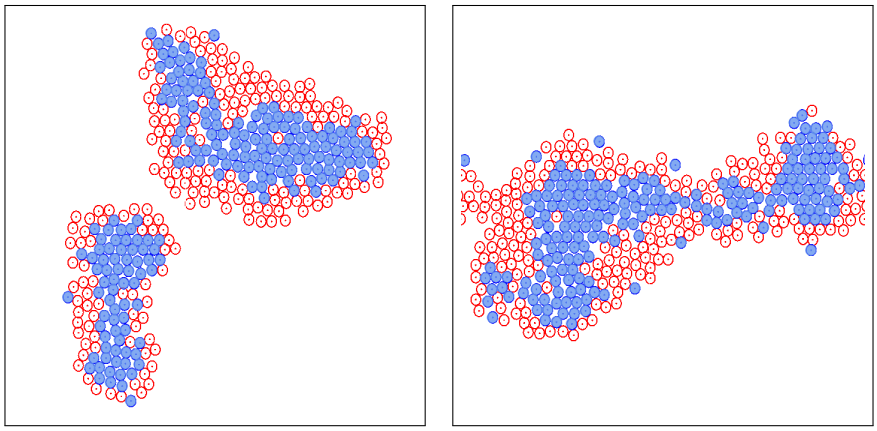
form the central core, the speed of the red agents simply determines the time at which the final sorted configuration is achieved.

T_S

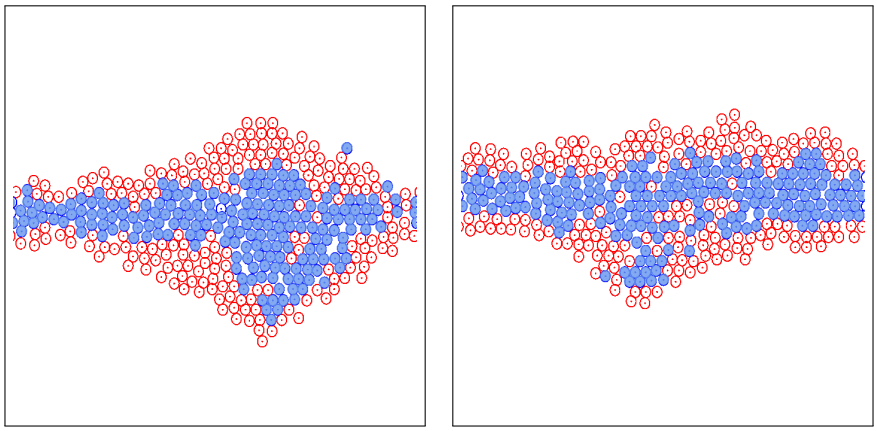
T_S is the elapsed time between when a blue T_1 agent makes its first attachment and when it begins C_2 production. T_S hours after blue agents begin to attach to each other, red T_2 agents, in response to increasing levels of C_2, begin to move towards blue agents, causing the red agents to enclose the blue aggregate(s). After several simulations we observed that $T_S = 18$ hours gives the best results. Figure 2.10(b) presents the result when this time is reduced to 6 hours. A premature aggregation of red T_2 agents prevents the blue agents from forming a single, symmetric core. Increasing T_S beyond 18 hours does not change the final desired sorted aggregate, it just lengthens the time to produce this result.

P_{Attach}

The probability of attachment P_{Attach} is a function of the types of agents that come into contact, as seen in Equation 2.7. In order to explore the role of attachment probability during agent sorting, simulations were performed where the probability of blue-blue (B-B), blue-red (B-R) and red-red (R-R) attachments were given all combinations of 10%, 50% and 100%, producing 27 agent sorting results. Most combinations of these probabilities produced some reasonable form of agent sorting. Interestingly, setting the probability of R-R attachments to 100% usually disrupted the desired agent sorting configuration, as seen in Figure 2.11. With attachment probabilities of B-B = 50%, B-R = 100%, R-R = 100% the sorting algorithm produced two approximately sorted structures, as in Figure 2.11(a). Attachment probabilities of B-B = 50%, B-R = 50%, R-R = 100% further disrupt the agent sorting process, producing a singly connected, somewhat chaotic strand in Figure

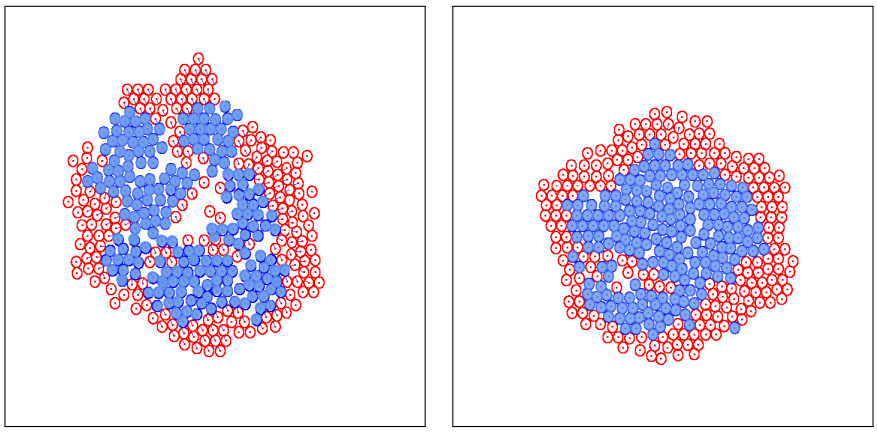


(a) quality=0.56 **(b)** quality=0.59



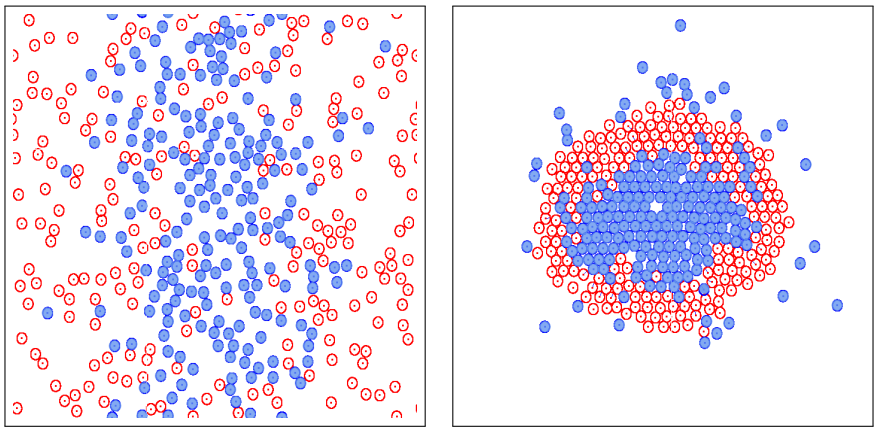
(c) quality=0.68 **(d)** quality=0.61

Figure 2.11: Effect of P_{Attach} on sorting, ($P_{Attach}(blue - blue)$, $P_{Attach}(blue - red)$, $P_{Attach}(red - red)$). (a) (50%, 100%, 100%); (b) (50%, 50%, 100%); (c) (100%, 10%, 100%); (d) (100%, 100%, 100%).

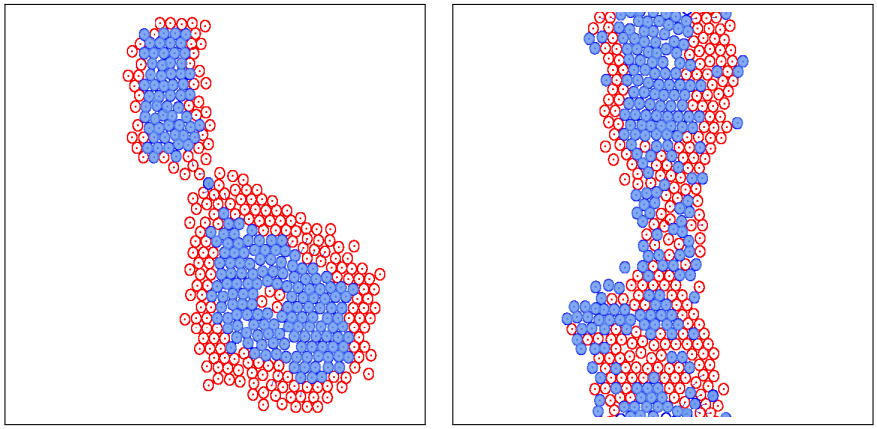


(a) quality=1.12 **(b)** quality=1.42

Figure 2.12: Effect of P_{Detach} on sorting. (a) $P_{Detach} = 0$; (b) $P_{Detach} = 0.1$.



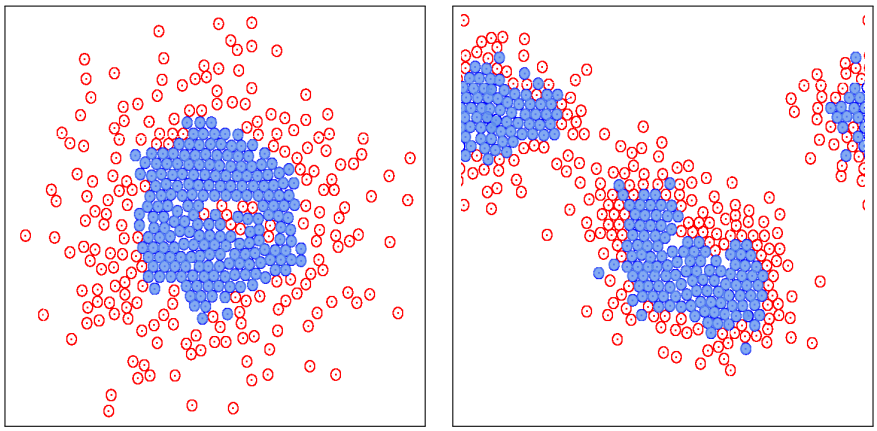
(a) quality=0.58 **(b)** quality=1.09



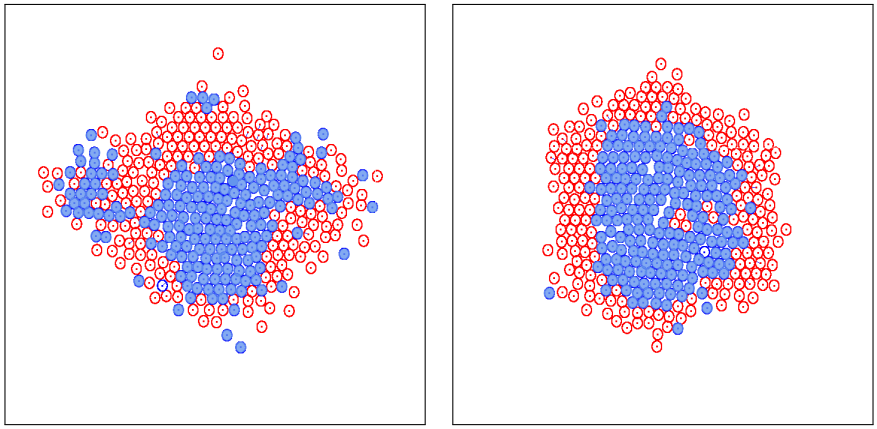
(c) quality=0.56 **(d)** quality=0.63

Figure 2.13: Effect of changing P_R for T_1 agents (blue). (a) $P_R = 10\%$; (b) $P_R = 20\%$; (c) $P_R = 80\%$; (d) $P_R = 100\%$.

2.11(b). Attachment probabilities of B-B = 100%, B-R = 10%, R-R = 100% produces another slightly-ordered strand-like structure in Figure 2.11(c). When all collision events produce attachments between agents (B-B = 100%, B-R = 100%, R-R = 100%) an even more uniform strand is produced, as in Figure 2.11(d). Note that since the simulations are performed in a toroidal environment the left edges of these images are connected to the right edges. All of these examples highlight the importance of weak R-R attachments when producing the desired sorted result. If red T_2 agents strongly attach to each other they prematurely form red aggregates that interfere with the aggregation of the blue T_1



(a) quality=1.03 **(b)** quality=0.46



(c) quality=0.96 **(d)** quality=1.41

Figure 2.14: Effect of constant P_R for T_2 agents (red). (a) $P_R = 10\%$; (b) $P_R = 30\%$; (c) $P_R = 50\%$; (d) $P_R = 75\%$.

agents. The blue agents need to aggregate freely with each other. The red agents need to stay unattached until they engulf the solid blue core later in the simulation.

P_{Detach}

Defining P_{Attach} by Equation 2.7 and setting P_{Detach} to 0, i.e. once an attachment is made it is never broken, produces a less desirable sorted result. In the scenario of Figure 2.12(a), the permanent attachments formed during random collisions trap red T_2 agents inside the blue core, and also prevent the agents from collapsing into a single mass; thus producing

interior holes. Allowing some detachments, as Figure 2.12(b), where $P_{Detach} = 0.1$, does improve the result, but still produces some holes and trapped red agents.

P_R

The probability that an agent will follow the gradient of a chemoattractant chemical is defined as P_R. If the probability of gradient following is too low no aggregation is observed, as seen in Figure 2.13(a). As the probability increases blue T_1 agents start forming a single aggregate. A tightly coupled aggregate is formed when P_R is 20%, as in Figure 2.13(b). It can also be seen in this figure that some blue agents never connect to the main blue aggregate because of their diminished response. The desirable aggregate shape is produced when this probability is between 40% and 70%. Our experiments showed that an increase in T_1's P_R from 0.7 to 0.8 significantly changes the agent sorting behavior. Once this parameter is over 0.7, T_1 agents do not come together into a well-formed single symmetric structure. For P_R equal to 80%, as in Figure 2.13(c), two separate blue aggregates are formed. When P_R is set to 1, i.e. there is no randomness in the chemotactic response, T_1 agents clump into a somewhat chaotic elongated structure, as seen in Figure 2.13(d). This highlights the need for noise and randomness in the sorting process in order to ultimately achieve an ordered final structure. If there is no noise in the movement of the agents, the randomness of the initial conditions is incorporated into the final sorting result. Once again recall that our simulations are conducted in a toroidal environment, so that the top of this long structure connects to the bottom.

The effect of a *constant* P_R value for T_2 agents was also studied, instead of using a gradually increasing function based on the local chemical concentration, as in Equation 2.5. Figure 2.14 shows the effects of varying the T_2 agents' constant P_R value on the structure of the final sorted agents. Setting $P_R = 10\%$ produces a result where red agents form a loosely

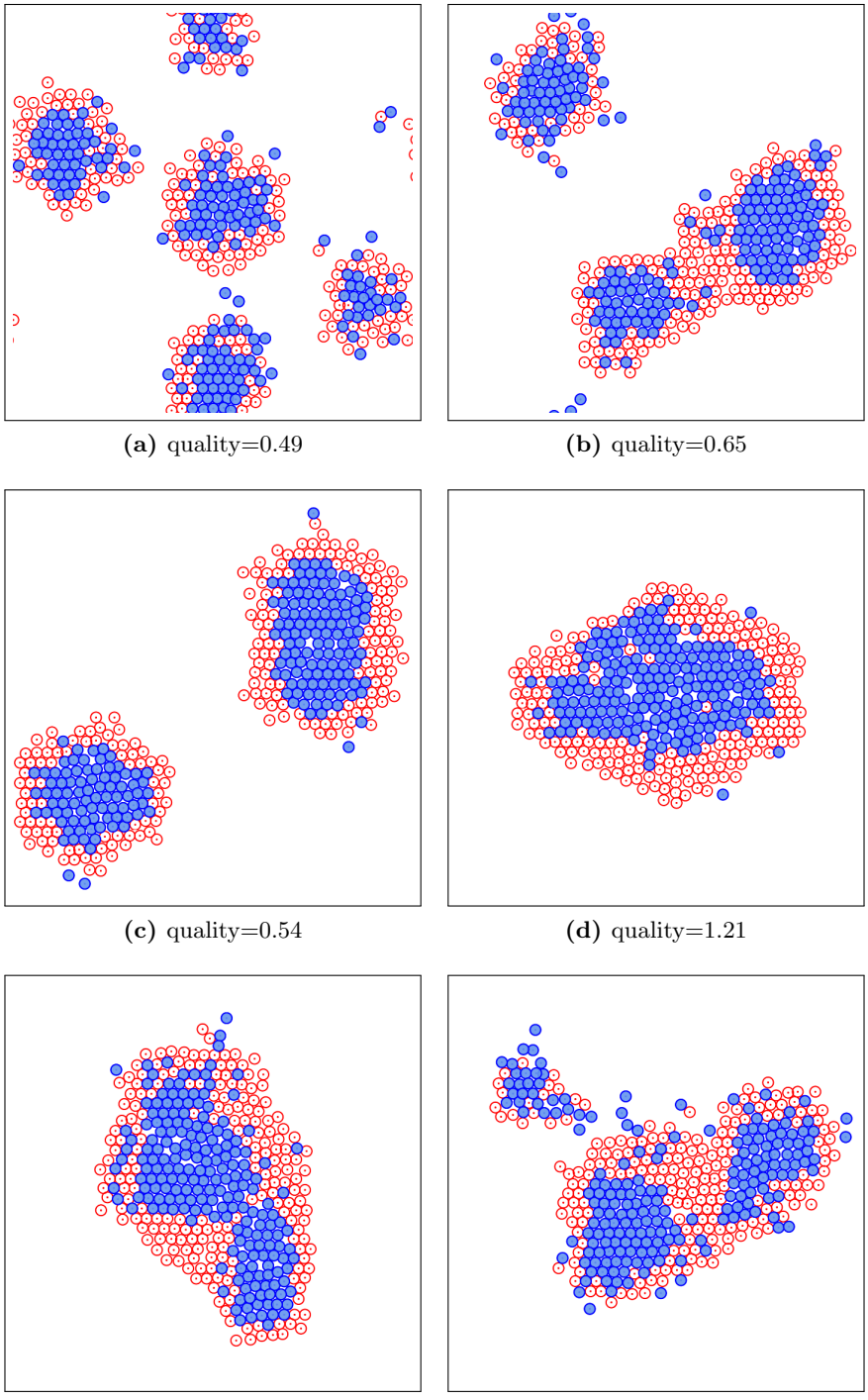
packed cloud around the blue agents. See Figure 2.14(a). Once this parameter is over 10%, T_2 agents aggregate more strongly and begin to surround the blue T_1 agents. At P_R equal to 30% the T_2 agents' behavior interferes with the aggregation of the T_1 agents, and the T_1 agents remain in separate groups, rather than aggregating into a single large structure. Since the response stays constant (and relatively weak) throughout the simulation these smaller aggregates stay isolated from each other rather than forming one single aggregate. See Figure 2.14(b). Strengthening the T_2 agents' aggregation ($P_R = 50\%$), as in Figure 2.14(c), brings the separate groups into a single, mixed aggregate. Further increasing P_R for T_2 agents to 75% and higher, as in Figure 2.14(d), brings more order to the aggregation and produces a barely acceptable sorted result.

R_{Max}

R_{Max} defines the radius of the chemical field emitted by one agent that can be sensed by other agents. In other words, if two agents are within the distance R_{Max} they are able to interact with each other via their chemical fields. Changing R_{Max} influences the final sorted structure. If R_{Max} is not large enough, instead of forming one single sorted structure the two types of agents form multiple sorted structures, as seen in Figures 2.15 (a), (b) and (c). However, if R_{Max} is too large, more and more agents begin to interact with each across the toroidal boundaries of the environment, which disrupts the sorting process, as seen in Figures 2.15 (e) and (f). R_{Max} values that are close to the optimal value of 300, e.g. 250 (as in Figure 2.15(d)), 350 and 400, produce nearly acceptable sorting results.

Acceptable Range of Parameter Values

Analyzing the results of the studies of the agent parameters, we identified the ranges of parameter values that produce the desired, sorted configuration and summarize them in



(a) quality=0.49 **(b)** quality=0.65

(c) quality=0.54 **(d)** quality=1.21

(e) quality=0.88 **(f)** quality=0.65

Figure 2.15: Effect of changing R_{Max} for both T_1 and T_2 agents. (a) $R_{Max} = 100$; (b) $R_{Max} = 150$; (c) $R_{Max} = 200$; (d) $R_{Max} = 250$; (e) $R_{Max} = 450$; (d) $R_{Max} = 500$.

Table 2.3. The desired sorted structures produced from utilizing these parameter values have a quality value above 1.40.

Table 2.3: Parameter value ranges that produce desired sorted structure.

Type	λ_i	P_R	P_{Detach}	P_{Attach}	T_S	R_{Max}
T_1	≥ 3	0.4 – 0.7	≥ 0.1	-	≥ 9 hrs	300 - 325
T_2	> 0.0	≥ 0.75	≥ 0.1	R-R: < 1.0	-	300 - 325

These parameter value ranges may be better understood within the context of a robot swarm application. For this application, distances can be measured in centimeters and time can be measured in seconds. Robots would not interact via chemical signals, but rather with radio signals that communicate the distance between a pair of robots. The robots would have the ability to attach to each other. A single robot would have a diameter of 12 cm, and its maximum velocity would be 1cm/s. The parameter values of Table 2.3 are valid for 200 T_1 (blue) robots and 200 (red) T_2 robots, initially mixed in a 500 cm $\times$ 500 cm arena. These values will produce a single sorted structure, where the T_1 (blue) robots form a central core surrounded by a layer of (red) T_2 robots, as seen in Figures 2.4 and 2.9.

The λ parameters are scale factors used to determine a robot's velocity, given the gradient of the cumulative "chemical" concentrations in the environment (Equation 2.1). A λ_1 value of 3 or greater will guarantee that the blue robots are fast enough to aggregate into a central core before time T_S, the time when the blue robots send out a signal that attracts the red robots. Within a cell context, where the blue agents' speeds are determined by the value of λ_1, it was found that T_S needed to be at least 9 hours. This value gave the blue agents sufficient time to aggregate. In the robotics scenario, the time units would be minutes rather than hours. It was found that the speed of the red robots did not affect the final outcome of the sorting, as long as it was non-zero. The value of λ_2 only affects the

time needed to produce the final, desired configuration.

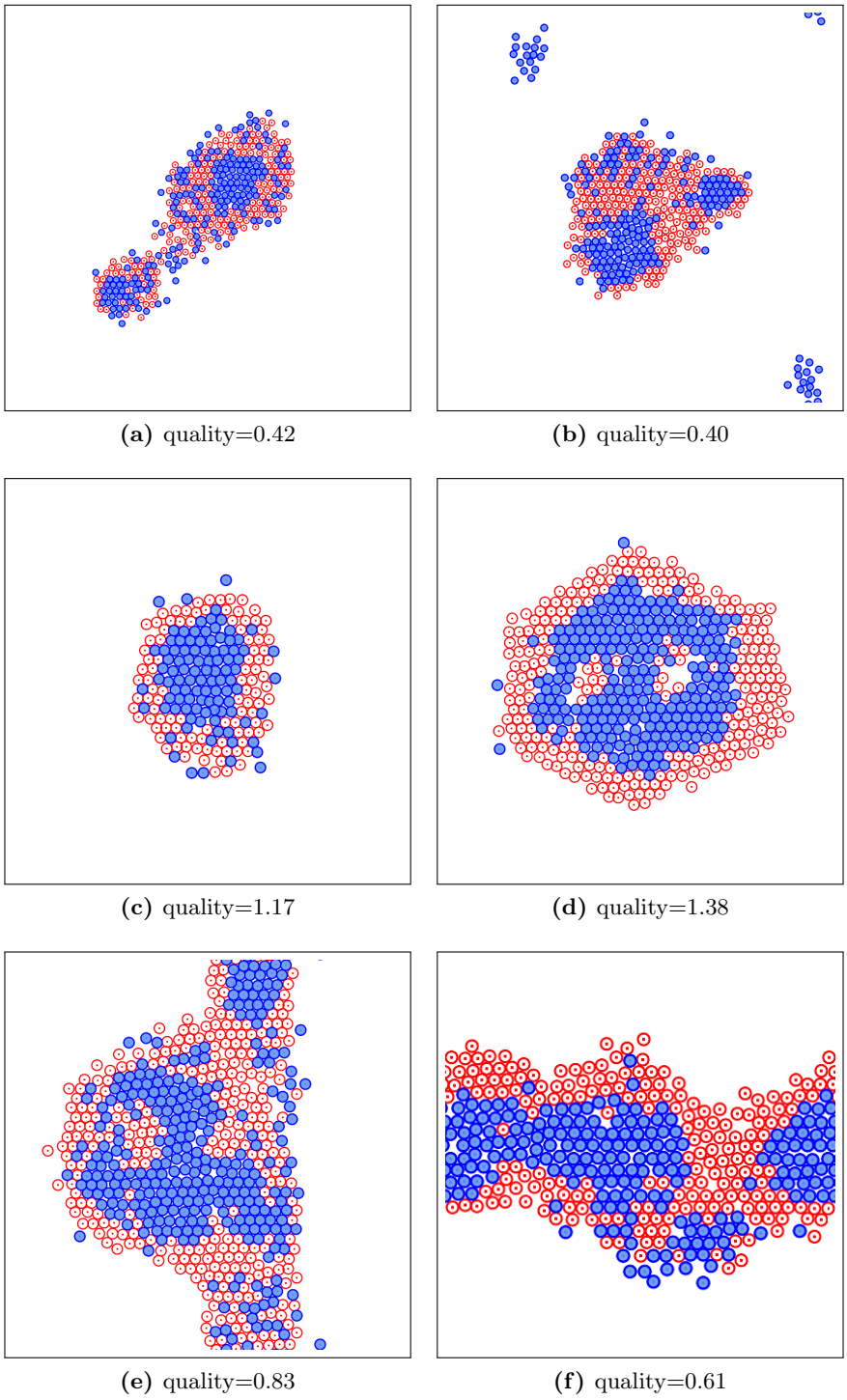
P_R, the probability that a robot will respond to a "chemical" signal, needs to have different values for the T_1 and T_2 robots. Values between 0.4 and 0.7 provide enough randomness in the T_1 aggregation process to allow the robots to "shake" into the desired central blue core. Once this core is formed, the T_2 robots need a strong and mostly deterministic ($P_R \geq 0.75$) response to their attracting signal in order to create a tight and confining layer around the blue core.

The value P_{Detach}, the probability that robots will detach from each other, has a small lower bound (0.1). We also found that P_{Attach}, the probability that two robots will attach to each, only significantly affects the sorted outcome when the probability that one red robot attaches to another red robot is 100%. This situation tends to disrupt the sorting pattern. Since we produce acceptable sorting results for a very wide range of P_{Detach} and P_{Attach} values, an attachment capability may not be necessary for robots in a swarm application, a point that requires further study.

A robot will only interact with another robot, i.e. exchange distance information, if the robot is within a distance of R_{Max}. We found that in order to produce a single sorted structure with 400 simulated robots within a 500 cm $\times$ 500 cm arena R_{Max} needed to be between 300 and 325 cm. If R_{Max} is lower than this range, multiple sorted structures are produced. Our parametric studies highlight the fact there is a complex relationship between arena size, robot density and R_{Max} that warrants further study. See Section 2.5.3 for additional comments on this issue.

2.5.3 Environment Parameters

In order to better understanding the influence of global parameters on agent sorting, we explored the parameter space associated with the system level of our simulations, such as



(a) quality=0.42 **(b)** quality=0.40

(c) quality=1.17 **(d)** quality=1.38

(e) quality=0.83 **(f)** quality=0.61

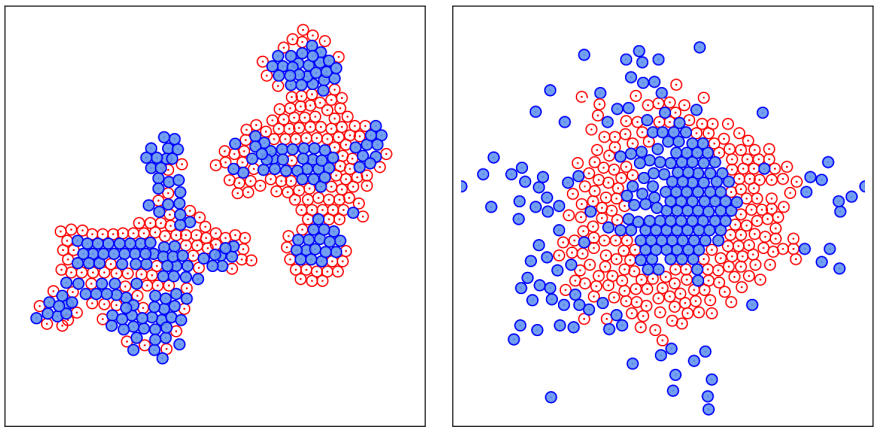
Figure 2.16: Effect of agent density. (a) 800^2 grid, 400 agents, density: 0.0006 agents/unit^2; (b) 700^2 grid, 400 agents, density: 0.0008 agents/unit^2; (c) 500^2 grid, 200 agents, density: 0.0008 agents/unit^2; (d) 500^2 grid, 500 agents, density: 0.0020 agents/unit^2; (e) 500^2 grid, 600 agents, density: 0.0024 agents/unit^2; (f) 400^2 grid, 400 agents, density: 0.0025 agents/unit^2.

agent density (by changing the size of the agent population and the computational arena), noise in the system and the initial distribution of the agents. While fixing agent parameters to be the ones in Table 2.2, we modified environment parameters one at a time in this study.

Agent Density

The density of agents in the computational arena may be varied by either changing the total number of agents or changing the size of the arena. We have conducted both of these experiments and a sampling of the resulting sortings are presented in Figure 2.16. When the density is low and the agents are spread apart from each other, the blue agents are unable to come together into a central core, either because the distances between agents prevent interaction or because the agents don't have enough time to aggregate before the red T_2 agents attempt to enclose the blue T_1 agents. This can be seen in Figures 2.16 (a), (b) and (c). Here the densities are 0.0006, 0.0008 and 0.0008 agents per unit squared. In Figures (a) and (b) density is lowered by increasing the size of the arena. It can be seen in Figure (b) that the blue agents formed three separate clumps. For comparison, the agent density for the "optimal" result in Figure 2.9 is 0.0018 agents/unit^2.

As density increases to near the optimal value a sorted structure with a high quality value emerges, as in Figure 2.16(d). Further increasing density, by increasing the number of agents as in Figure 2.16(e) and decreasing the size of the arena as in Figure 2.16(f), disrupts the sorting process. The toroidal structure of the computational environment leads to the agents forming a single, connected aggregate. In Figure 2.16(e) a slightly sorted structure is formed. Increasing the density in Figure 2.16(f) significantly degrades the quality of the sorting.



(a) quality=0.60 **(b)** quality=0.76

Figure 2.17: (a) Structure formed after decreasing the magnitude of noise during movement. (b) Increasing the magnitude of the movement noise disrupts the sorting process.

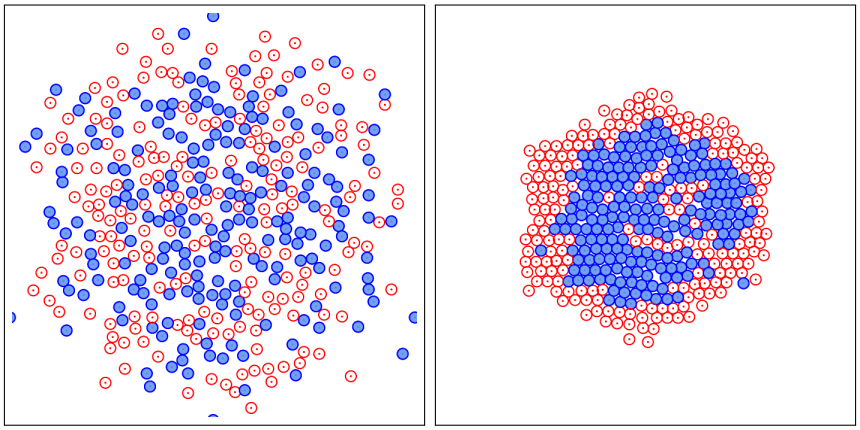
Noise

If an agent does not detect a chemical field or if it collides with another agent without attaching, the agent makes a random movement during that time step of the simulation. Also, all agents do not always follow a detected chemical gradient, they take random steps with a certain probability. The magnitude of the random movement has an impact on the final sorted structure. For example, when the step size is too small, e.g. 1 unit per random movement during a single time step, agents are unable to sort properly, as seen in Figure 2.17(a). There is insufficient noise in the system for the agents to achieve the global minimum of the well-sorted structure, to use a Metropolis algorithm [70] analogy. The initial randomness of the system is evident in the final result. Recall that we found the optimal magnitude of the random step to be 1 to 6 units, with the radius of an agent being 6 units. Increasing the magnitude of the random step to be between 6 and 12 units also prevents proper sorting, as seen in Figure 2.17(b). Note the probability of taking a random step for blue agents is constant at 50% and is a decreasing function of the C_2

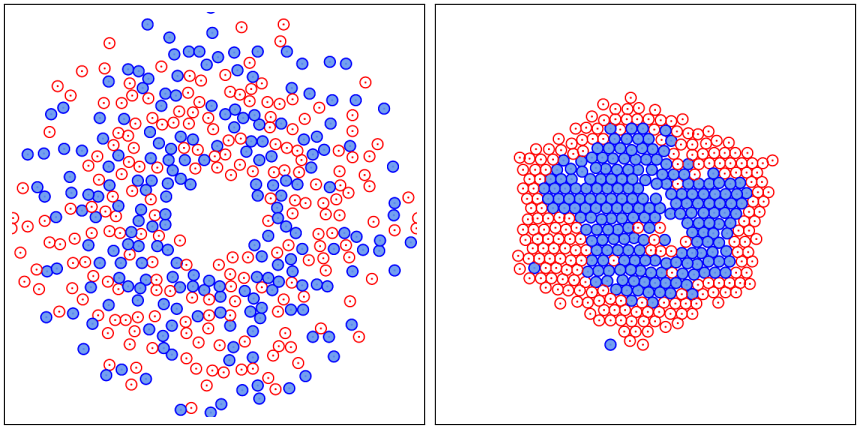
concentration for red T_2 agents. This explains why the red agents have mostly formed into a single encircling mass, while many of the blue agents create a "cloud" on the outside of the aggregate. Red agents are less affected by the magnitude of the random step, since in general they make fewer random movements.

Initial Conditions

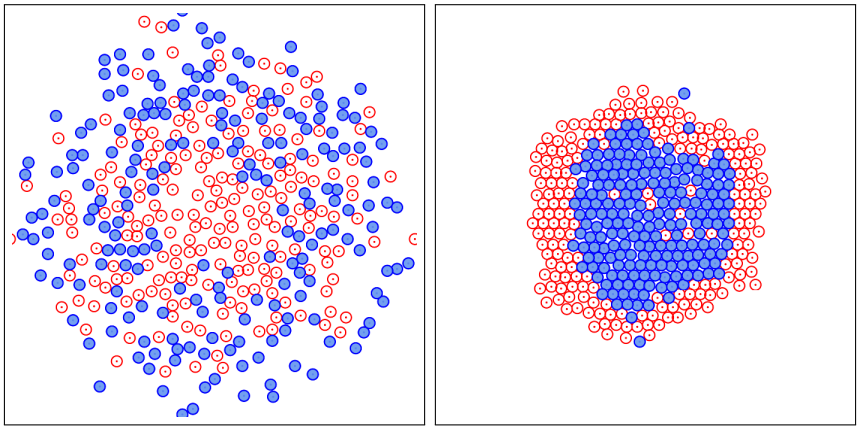
All previous sorting simulations began with a uniform random distribution of the agents' positions. In this set of experiments, we explored the effect of utilizing different distributions when initializing the positions of the agents. Three experiments were performed. The first used a Gaussian distribution with mean 0 and standard deviation (σ) of 1. The distribution was sampled up to 3σ to produce starting locations for the agents, as seen in Figure 2.18(a). The second experiment used a Landau distribution that was sampled in the range -6 to 10. A typical initial condition from this distribution is presented in Figure 2.18(b). In the final experiment we used the Gaussian distribution for the red agents and the Landau distribution for the blue agents, as seen in Figure 2.18(c). Utilizing the distributions in this manner generates initial conditions that are opposite the desired result, with the red agents predominately near the center of the environment and surrounded by blue agents. 100 agent-sorting simulations were performed with each of the initial conditions. Out of the 300 simulations only one had a quality measure above 1.4. This was produced from the mixed Gaussian/Landau distribution, and is presented in Figure 2.18(c). In fact, most of the simulations produced unacceptable sorting results, with the average quality measure for the Gaussian distributions being 1.09, 1.15 for the Landau distributions, and 0.919 for the mixed Gaussian/Landau distributions. The best results for the other two distributions are also presented in Figure 2.18, with the Gaussian distribution having a top quality value of 1.27, and the Landau distribution having a top quality value of 1.36. With one exception,



(a) Gaussian initial conditions, quality=1.27



(b) Landau initial conditions, quality=1.36



(c) Mixed Gaussian/Landau initial conditions, quality=1.49

Figure 2.18: Initial agent distributions and final, sorted results for a variety of initial conditions. (a) Gaussian distributed initial condition, best sorted result; (b) Landau distributed initial conditions, best sorted result; (c) Red T_2 agents with Gaussian distributed initial condition, blue T_1 agents with Landau distributed initial conditions, best sorted result.

we have only been able to produce the highest quality sorting results (with a quality measure above 1.4) with simulations that have an initial uniform random distribution of the agent locations. These results demonstrate that bias in the initial conditions disrupts the cell sorting process, and that uniform random initial conditions are needed to produced the desired, sorted outcome.

2.6 Conclusion

We have presented a 2-D chemotaxis-based algorithm that robustly creates a layered, sorted structure from heterotypic self-organizing agents. Our self-sorting agents may be considered truly self-organizing because they consist of properties and behaviors not exhibited by related systems. Unlike previous work [72, 73, 108, 28], our agents agents do not know their global position, do not contain a representation of the shape to be produced, are not assigned a location in the final configuration, only interact with neighboring agents, base their actions on local and internal information. In our system, agents of the same type (T_1 or T_2) execute the same program, are not directed by a centralized controller and therefore do not act to minimize a globally evaluated function. We are able to achieve a self-sorted result by following a chemotaxis paradigm inspired by cell biology. Agents emit chemicals into the environment and follow gradients of the accumulated chemical field, as well as perform prescribed actions like probabilistic attachment or detachment.

We have designed an evaluation mechanism to quantify the fitness/goodness of the final sorted structure. We have identified the response magnitude to chemoattractant λ, probability of responding to a chemoattractant P_R, probability of detachment P_{Detach}, probability of attachment P_{Attach}, and time between T_1 agent's first attachment and production of chemoattractant C_2, i.e., T_S to be the important parameters influencing the sorting process. Simply put, λ controls how fast a cell moves proportional to the chemical gradient it

senses on the surface, therefore, λ controls how fast agents aggregate. P_R introduces noise in the system, which may enhance the system's ability to find an optimal configuration. P_{Detach} and P_{Attach} control the shape of the aggregate. T_S controls when T_1 agents begin to attract T_2 agents. A proper combination of these parameters ensures the formation of the structure of the final sorted aggregate. Future work will involve interpreting the meanings of the parameter ranges and exploring their biological significance. We have shown that via parametric analysis, we are able to propose parameter ranges that robustly produce a sorted structure with fitness value above a certain threshold.

While this process is robust, it only produces one specific sorted configuration, which is achieved by changing parameters of a fixed set of cell actions. Additionally, in this system, agents move on an underlying hexagonal grid, which causes the final structure to display a macroscopic hexagon shape. In order to remove this underlying bias and to form arbitrary shapes, we eliminate the hexagonal grid and allow agents to move in a continuous $2D$ space. The ultimate goal of this research is to produce a more general robust self-organizing system that is capable of generating different types of macroscopic structures. Therefore, we extend our biologically inspired agents to become general shape formation primitives whose interactions are automatically discovered with genetic programming.

Chapter 3: General Shape Formation Primitives

3.1 Introduction

Motivated by the ability of living cells to form specific shapes and structures, we extend our previously developed cell sorting model, described in Chapter 2, to be a more general self-organizing shape formation system for chemotaxis-inspired cellular primitives. Our cell sorting system consistently forms a final sorted structure utilizing fixed local interactions between agents. However, we aim to enhance the system to form various user-defined shapes and structures. We modify the fixed local interactions in our cell sorting system to be an artificial interaction function defined as a mathematical expression. We successfully utilize genetic programming to evolve the local interaction functions for a number of user-predefined macroscopic shapes.

In this work, virtual cells are first placed into a 2D environment with a random uniform distribution, and the cells interact with each other via a chemotaxis paradigm. This interaction produces movements that lead them to aggregate into a single user-specified shape. We call these self-organizing virtual agents morphogenetic primitives (MPs). Each MP is represented by a small disk and emits a 'chemical' which accumulates in the environment. An MP detects the cumulative field at eight receptors on its surface, and calculates the field gradient from this input. MPs move in the direction of the field gradient with a speed proportional to the magnitude of the gradient. As seen in Figure 3.1, by employing these relatively simple chemotaxis-inspired behaviors MPs are able to self-organize into macroscopic shapes; thus providing a distributed capability that could be useful for swarm and/or reconfigurable robots.

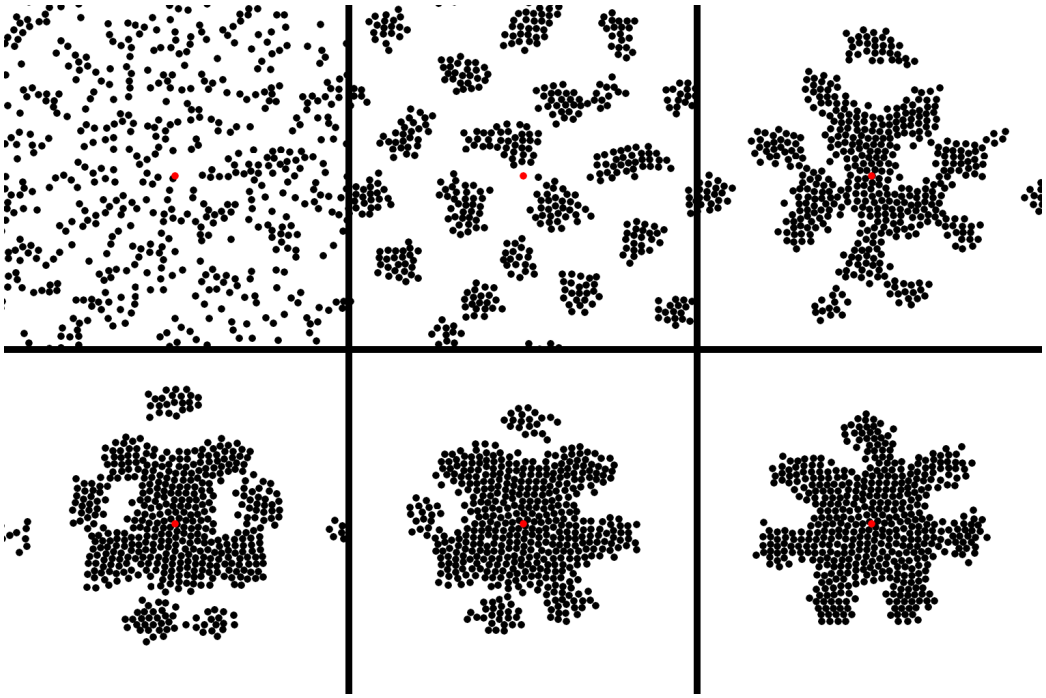


Figure 3.1: Morphogenetic primitives self-organizing into a gear shape.

While MPs' fundamental interaction is based on a chemotaxis-inspired paradigm, we do not limit their behaviors/properties to be physically realistic or completely consistent with biology. Instead, developmental biology provides a motivating starting point for MPs. As a way to customize chemotaxis-inspired cells for shape formation, we alter the chemical concentration fields around individual cells. Instead of the chemical concentration dropping off as a function of distance r ($1/r$, the physically accurate description), we define the concentration field with a mathematical function of cell-cell distance d, one cell's angular location θ in another cell's local coordinate system and simulation time t. The function set of the parameters is $F = \{+, -, *, /, exp, log, sin, cos\}$.

At this time, there is no prescriptive way to specify a particular local field function that will direct MPs to form a specific macroscopic shape. If we use a binary tree structure to represent the field functions, a tree with depth k has $O(4^n \cdot 8^n)$ variations, where $n = 2^{k-1}$,

4 is the cardinality of the terminal set $T = \{constant\ E, d, \theta, t\}$ and 8 is the cardinality of the function set F. Given the enormous size of the space of field functions that needs to be searched, we employ genetic programming [62] to produce the mathematical expression that explicitly specifies the field function. In order to meet the substantial computational requirement imposed by our evolutionary computing approach, we have implemented a master-slave form of the distributed genetic programming process. The fitness measure associated with each individual field function is based on the shape that emerges from the chemical-field-driven aggregation simulation, and determines which functions will be passed along to later generations. The genetic process stops once an individual (i.e., a mathematical expression) in the population generates the desired shape via a chemotaxis simulation, or after a certain number of generations have been produced and evaluated.

With this algorithm, we have successfully evolved chemical field functions for a number of simple shapes. See Figure 3.6 through Figure 3.9. These results furnish the evidence that supports the proposition that biological phenomena offer paradigms for designing cellular primitives for self-organizing shape formation. Additionally, evolutionary computing techniques, specifically genetic programming, have been crucial for discovering the detailed local interactions that lead to the emergence of macroscopic shapes and structures.

3.2 Related Work

Fleischer [44, 43, 45] created a cell-based multi-mechanism developmental model for studying emergent phenomena observed in biological systems, e.g. formation of axes, segmented patterns and hierarchical structures. He also applied his cell interaction simulation system to computer graphics to produce an approach to cellular texture generation [46]. Sims [90] describes one of the first applications of evolutionary techniques to computer graphics. Genetic programming is used to create functional representations of intricate, compelling

images and solid textures. This approach was extended to define 3D procedural models and parameters for dynamical systems [91], and to create the structure and behaviors of virtual creatures [92]. Eggenberger Hotz [33, 56] proposed the use of genetic regulatory networks coupled with developmental processes for use in artificial evolution and was able to evolve simple shapes. The combination of artificial evolutionary techniques and developmental processes provides a comprehensive framework for the analysis of evolutionary shape creation.

Theraulaz and Bonabeau [99, 100] present a modeling approach based on the swarming behavior of social insects, a type of swarm intelligence [15]. They combine swarm techniques with 3D cellular automata to create autonomous agents that indirectly interact in order to create complex 3D structures. This indirect interaction, known as stigmergy [101], allows the agents to act cooperatively, but independently, through a stimulus-response mechanism based on modifications made to the environment. Bonabeau et al. [16] apply genetic algorithms (GA) to the stigmergic swarm-based 3D construction method in order to direct the overall process.

Nagpal et al. [72, 73] present techniques to achieve programmable self-assembly. Cells are identically-programmed units which are randomly distributed and communicate with each other within a local area. In this approach, global to local compilation is used to generate the program executed by each cell, which has specialized initial parameters. Doursat [27, 29, 30, 31] proposed a model for artificial development which combines proliferation, differentiation, self-assembly, pattern formation and genetic regulation. Via genetic-like regulation at the agent level, the agents can self-organize into a number of patterned shapes and structures.

Beal [12] presents the concept of functional blueprints for grown systems. It is an approach for specifying a system in terms of desired performance and a means of incrementally

correcting deficiencies. By defining the deficiency as a lack of oxygen in a developing biological system, he demonstrates that a virtual vascular system may be dynamically created to maintain the functionality of the overall system as it develops and expands. Werfel [105] describes low-level primitives based on capabilities evident in biological systems, with the goal of developing a basis for engineering developmental processes that may be realizable with living cells. The primitives implement biologically realistic morphogen gradients to produce structures of desired size, provide positional information, and trigger genetic cascades that lead to the growth of more complex structures.

In the field of robotic swarms Stoy and Nagpal [95, 96] present an approach to self-reconfiguration based on directed growth, where the desired configuration is grown from an initial seed module. All modules in the robot are connected, and change from an initial configuration to a desired configuration, with the growth guided by a representation of the desired configuration. Shen et al. [88] proposed the Digital Hormone Model as a distributed control method for robot swarms. In one set of experiments, cells/robots are able to generate reaction-diffusion patterns on a grid in a distributed fashion. In another set of experiments, robots are attracted and aggregate around targets sensed over short distances. By electing leader(s) as barycenter(s), propagating gradients of varying structure and using these gradients as instruction conditionals, a swarm of simulated robots developed by Mamei et al. [68] are able to self-organize into a number of simple shapes such as a circle, ring, lobes and polygons. Swarm chemistry, proposed by Sayama [81, 82] and based on Reynold's boid model [78], is an approach for designing spatio-temporal patterns for kinetically interacting, heterogeneous agents. An interactive evolutionary method has been used to define system parameters that lead to agent segregation and structure formation.

Werfel and Nagpal [106, 107, 108] present algorithms that employ extended stigmergy

for the assembly of solid structures. The elements are separated into mobile and structural components (i.e., robots and modular blocks, which are the same shape but have different functions). The robots communicate indirectly through information that is stored in the blocks and transferred during attachment. The robots and blocks are placed in an obstacle-free workspace, along with a beacon indicating the starting location of the construction. The robots collect blocks from caches and arrange them into a structure. A block can immediately obtain, from its neighboring blocks, information about its global position and the present and desired final structure.

Our work is similar to previous work in that 1) we make use of a chemotaxis-inspired cell aggregation model; 2) we apply evolutionary computing to discover interaction rules; and 3) we develop an approach for multi-agent shape composition. However, our approach is novel compared to previous work in that it contains all of the following features. 1) All morphogenetic primitives are randomly placed in the environment, are identical, and perform the same simple actions. They require no special seed primitives or customized initialized states. 2) No initialization of spatial information is needed in the computational environment. We utilize a toroidal environment where all positions are considered identical and play no special role. 3) Individual MPs do not know their location in any external/global coordinate system, and deposit no unique information at specific locations in the environment. 4) MPs do not contain a representation of the predefined global shape that is being composed. 5) Genetic programming is used to explore the space of concentration field functions that surround the individual primitives. Automatic fitness evaluation is implemented in our method, which calculates how similar an aggregate is to the final shape and produces a scalar fitness value. Genetic programming uses this fitness value to generate a field function that directs the individual primitives to form into a user-specified shape.

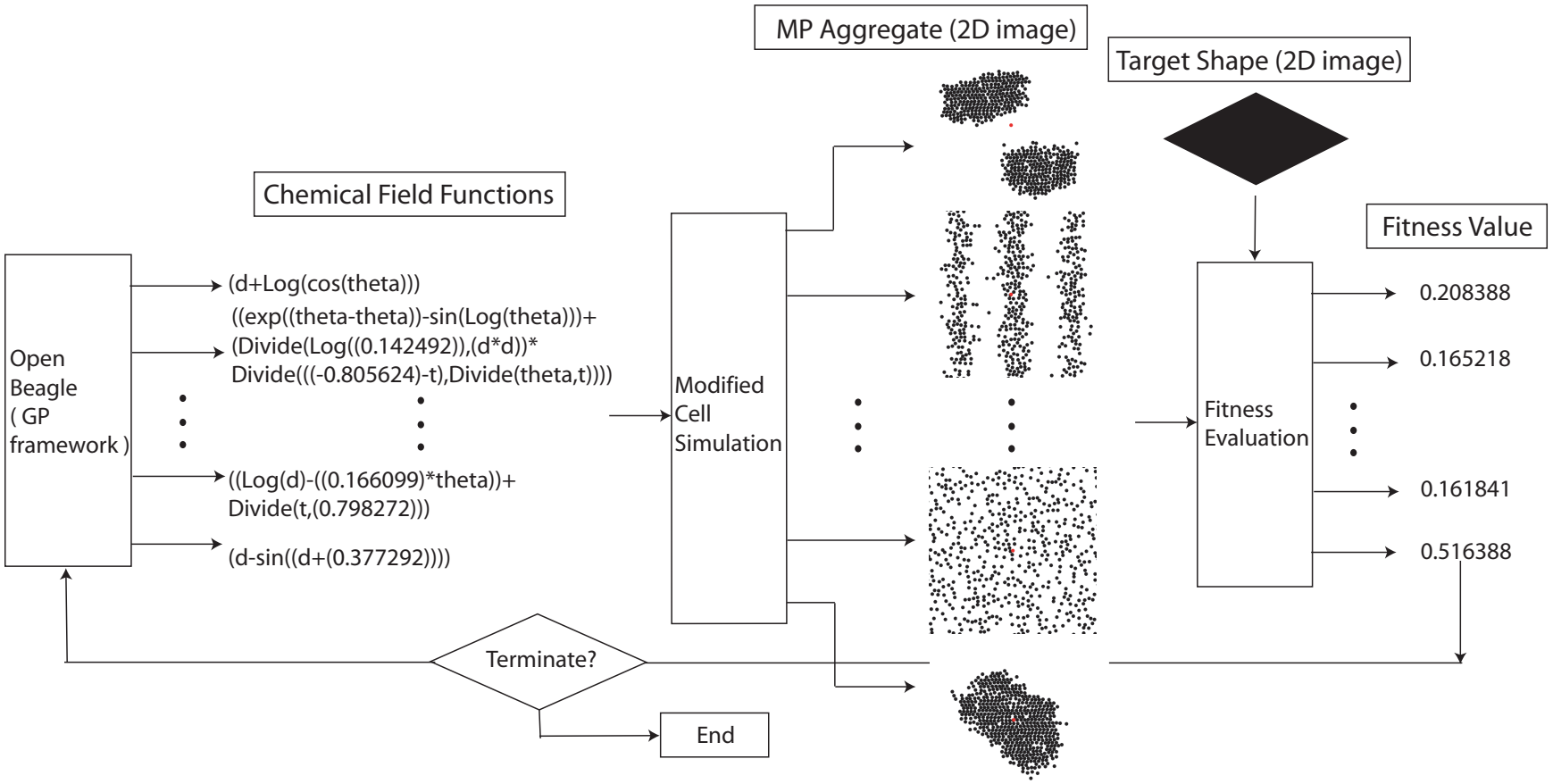


Figure 3.2: Overview of the genetic programming process that produces the chemical field functions of morphogenetic primitives.

3.3 Approach Overview

There are two major components in the self-organizing shape formation process: an MP aggregation simulation system and a genetic programming framework. Using chemotaxis as a paradigm, the aggregation simulation system calculates the movements of each primitive during the spatial self-organization process. The genetic programming (GP) framework is used to discover the local interaction rules employed during the computed simulation. It is these rules that direct MPs to aggregate into user-specified shapes and patterns. The local interactions rules are explicitly represented as 'chemical' field functions with variables of cell-cell distance d, one cell's angular location θ in another cell's local coordinate system, and simulation time t, i.e., $f(d, \theta, t)$. In the context of a GP framework, the 'chemical' field functions are the individuals to be evolved [7].

The general approach, which has been implemented within the Open Beagle Framework [48], to defining the field functions that ultimately produce the user-desired shape is presented in Figure 3.2. We start with a population of functions that is initially randomly generated. Each function is compiled into a chemotaxis-based cell aggregation simulation, and defines the chemical field that surrounds all of the individual cells for a particular aggregation simulation. A chemotaxis simulation is then performed. Each MP simulation usually produces some kind of aggregated structure. The resulting MP configuration (the simulated aggregation) is compared to the user-desired shape, and a scalar fitness value is calculated that quantifies how well the computed shape matches the desired shape. A subset of the top candidates is then used to create the next generation of field functions. The process continues until a field function produces the desired shape or the maximum number of generations is reached.

We have utilized this new approach to define morphogenetic primitives that aggregate to

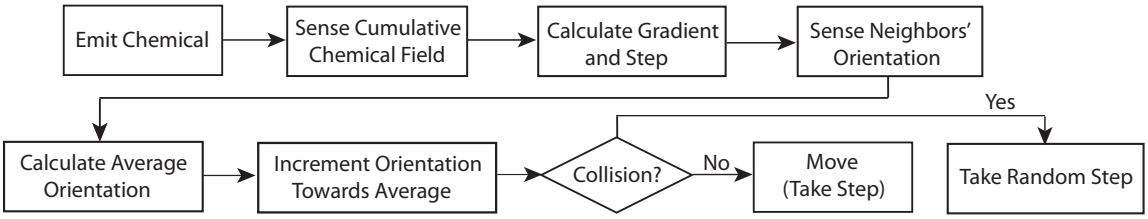


Figure 3.3: Overview of the steps executed in the chemotaxis simulator.

form a number of user-defined shapes, e.g., an ellipse, a diamond, an hourglass, a cross, the letter 'b', the letter 'S', a star and a triangle. In the process of evolving the field functions for user-defined shapes, a few pleasant surprises emerged. Most of them included repeated patterns, e.g., stripes, spots and sine waves, but interestingly a gear shape emerged from the evolutionary process. We believe that this new approach to automated shape formation can be extended and applied to a number of applications, including the control of robotic swarms, motion specification of animated crowds, and generative model creation.

For example, when applying this approach to robotic swarms, each MP is now an autonomous robot that broadcasts a weak signal into its immediate neighborhood, in order to detect its neighbors. Robots can only sense signals emitted by other robots within a certain distance, because of the weakness of the signal. Neighboring robots respond to the detection signal, and from this the transmitting robot is able to produce a neighbor list. Once a robot has a neighbor list, it communicates with its neighbors to get distance and orientation information, which can be used to calculate the concentration values needed for its local movements. Concentration gradients can be calculated by having the robots take small random steps and reacquire concentration information. Detection signals are sent out at regular intervals in order to maintain a correct neighbor list.

3.4 Morphogenetic Primitives

3.4.1 2-D Biological Cell Aggregation Simulation

MPs are based on a computational model and software system that is capable of simulating chemotaxis-based cell aggregation in 2-D [38, 40]. This 2-D chemotaxis-based cell aggregation model incorporates fundamental parameters involved in cell-cell aggregation. Using appropriate approximations / assumptions with efficient and effective algorithms, the model forms the core of a robust and extensible simulation environment, which has enabled us to study and identify the critical components of cell interaction and behavior that most influence aggregation outcomes.

In this earlier system, each cell is defined by a collection of physiologically relevant parameters and actions, such as the location of the cell, age, life cycle stage, chemoattractant secretion and response rates, diffusion radius, proliferation rate, adhesivity and number of attached cells. Virtual cells in the system are able to emit chemoattractants, sense the chemoattractant gradient, move in the direction of the gradient, proliferate, adhere to other cells, age and die. The 2-D cell aggregation model was later extended to simulate self-sorting of heterotypic agents [37, 39]. The 2-D cell aggregation model provides a critical computational component for our morphogenetic primitive simulation framework.

Each MP aggregation process begins by randomly placing a number of MPs (500 for our examples) with uniform random positions and orientations in a periodic (i.e., toroidal) computational environment. In order to avoid outcomes influenced by boundary conditions, we make our computational environment toroidal. i.e. the top edge of our simulation arena is connected to the bottom edge, and the left edge is connected to the right edge. A single aggregation simulation is comprised of a series of time steps. For our current work we have disabled several of the features implemented in our earlier work, namely cell adhesion,

division and death. For each time step, each primitive performs a prescribed set of actions, which are outlined in Figure 3.3 and described in detail in the remainder of this section.

3.4.2 MP Design Principles

Several principles were followed when developing morphogenetic primitives. 1) MPs are autonomous 'agents'. Each MP is an independent entity that senses the environment, responds to it, and then modifies the environment and its internal state. There is no 'master designer' directing the actions/motions of the MPs. 2) Actions are based on local information. Each primitive emits a *finite* field that can be sensed only by other primitives within a certain range. The only information received by an MP is gathered at its surface, namely the concentration of the cumulative field and contact with immediate neighboring MPs. 3) MPs respond to information with prescribed behaviors. The actions performed by each MP are the same, but the specifics of the individual actions are based on information received from the environment. 4) While their actions are determined by the ultimate shape that they will produce, MPs do not carry or access information about the shape. 5) MPs do not know their current global position nor their final position relative to the specified shape. 6) The shape emerges from the aggregation of local interactions and behaviors. Rather than follow a plan to produce the shape, MPs sense, change and respond to the cumulative field concentration. This simple behavior, when combined with somewhat complex individual fields, will direct the MPs to take individual actions, based on local information, that will ultimately direct them to aggregate into a user-defined, macroscopic shape.

3.4.3 Movement

Similar to the behavior of cell-sorting agents, instead of implementing a complex form of chemotaxis-based motion [17, 59], we adopted a simplified chemotaxis-inspired method of

calculating MP movement. The 'chemical' gradient is used to determine the primitive's velocity. We assume that MPs travel at a terminal velocity through a viscous fluid environment, therefore an MP's velocity is directly proportional to the chemical field gradient (∇C). The calculation of the chemical field gradient is described in Section 2.3.1 and further defined in Equation 2.2. When an MP moves in the direction of the chemical gradient, its velocity ($\mathbf{V}$) is calculated as

$$\mathbf{V} = \lambda * \nabla C, \tag{3.1}$$

where λ (1 for our examples) is a constant that determines the magnitude of a cell's response to the gradient. At each simulation time step (Δt) the displacement ($\Delta \mathbf{x}$) of the MP is

$$\Delta \mathbf{x} = \mathbf{V} * \Delta t. \tag{3.2}$$

3.4.4 Rotation

Each MP is assigned its own local coordinate system with a random orientation ω between 0° to 359°, and is allowed to rotate. Additionally MPs are given the ability to detect the orientation of close neighbors and are programmed to rotate in a way that aligns their local coordinate system with the average orientation of immediate neighbors [6], a behavior observed in living cells [32].

In our simulations, 'nearby' neighbors are those MPs within a distance of four cell-radii. Each MP identifies its nearby neighbors, detects its neighbors' orientations, and calculates an average orientation $\hat{\omega}_{avg}$ via a vector average of the neighbors' orientations. To perform local alignment, MPs incrementally rotate one degree along the shortest arc towards the average orientation of its neighbors during each time step. Equations 3.3 through 3.6 detail

these actions.

$$\hat{\omega} = cos(\omega) + isin(\omega), \tag{3.3}$$

$$\hat{\omega}_{avg} = \frac{1}{n}\sum_{j=1}^{n}\hat{\omega}_j = x + iy \qquad j \in immediate\ neighbors, \tag{3.4}$$

$$\omega_{avg} = \begin{cases} acos(x/||\hat{\omega}_{avg}||) & y \leq 0, \\ 360^\circ - acos(x/||\hat{\omega}_{avg}||) & otherwise. \end{cases} \tag{3.5}$$

$$\omega_{new} = \begin{cases} \omega + (\omega_{avg} - \omega)/|\omega_{avg} - \omega| & |\omega_{avg} - \omega| \leq 180^\circ, \\ \omega - (\omega_{avg} - \omega)/|\omega_{avg} - \omega| & otherwise. \end{cases} \tag{3.6}$$

$\hat{\omega}$ is vector presentation of an MP's orientation ω and ω_{new} is the new orientation after the MP has rotated towards the average of its neighbors' orientations. These incremental rotations will eventually lead to the global alignment of all of the MPs' local coordinate systems [98].

3.4.5 Randomness in the System

Similar to stochastic optimization [84], randomness is injected into the system as a small amount of noise. Noise is important in preventing the set of MPs from forming into local minima configurations, and is included in our simulations in a number of ways.

When a displacement produced by following the chemical gradient, as in Equations 3.1 and 3.2, causes an MP to collide with another MP, a small step (1 to 6 units) is taken in a random direction with 90% probability, instead of the calculated step that produced the collision. After 1 out of 10 detected collisions, the MP does not move at all. If a second collision occurs after taking the random step, the primitive does not move and remains at

its current location.

MPs follow the chemical field gradient with probability P_{Follow}, which is a function of the current simulation time $t_{current}$ and end time t_{end} of the simulation, and is defined as

$$P_{Follow} = 0.5 * \left(1 - cos\left(\pi * \frac{t_{current}}{t_{end}}\right)\right). \tag{3.7}$$

This is a cosine function that goes from 0 to 1 over the span of the MP simulation. When an MP does not follow the chemical gradient, it takes a small random step. As simulation time increases the value of P_{Follow} increases, and fewer and fewer MPs make random movements; thus stabilizing the shape of the aggregate. This type of decreasing randomness is motivated by the annealing process of the Metropolis algorithm [61, 69].

Finally, in order to eliminate the side effects that might be associated with the organization of the internal data structures in our simulation system, MPs are processed in a random order at each time step of the simulation, rather than in the order in which they are stored. We have found that in general this improves the robustness and effectiveness of the approach. See Section 3.7.1.

3.5 A Genetic Programming Framework

3.5.1 Genetic Programming

Genetic programming (GP) is a type of evolutionary computation that provides an automated approach to software and function development based on genetics, reproduction, mutation, recombination, and selection. GP evolves a computer program or mathematical function based on its ability to perform a specified computational task, as evaluated by a fitness function [34, 62]. The underlying concept of evolutionary computation is: given a population of individuals and provided an evolutionary process of variation based on the

fitness of the individuals, produce a new population with higher fitness values than the previous population.

The evolutionary computation approach to a certain problem consists of the following components: individuals, fitness evaluation, selection and variation, initialization, a termination condition and control parameters. An individual is a representation in the form of a solution to the problem. For example, when GP is used to automatically generate computer programs, each individual of a GP generation is a parse tree that holds software instructions. Therefore, a population of individuals, which are going to evolve in the evolutionary process, represents a number of candidate programs that provide solutions to the problem. The fitness function defines how close a candidate solution is to the desired solution. A threshold value for this function defines the termination condition. In other words, fitness evaluation defines a similarity metric between an individual outcome and the final goal. The more similar the two are, the better (more fit) the individual is considered to be.

There are two kinds of selections: parent selection and survival selection. Parent selection or mating selection provides a higher probability that the higher fitness individuals will become parents in the next generation. While survival selection is a replacement strategy that swaps newly-generated individuals with older ones in order to increase the overall fitness of the new generation [34]. Selection is responsible for the quality improvement of the population. Variation, which includes reproduction, recombination and mutation procedures, is the mechanism that creates new individuals from the existing ones [62].

Genetic programming, as outlined in Figure 3.4, is different from other types of evolutionary computing algorithms in that the individuals are parse trees consisting of elements from a Function Set F and a Terminal Set T. Elements of T reside in the leaves of the parse trees and the symbols of F are stored in the internal nodes. The ramped half-and-

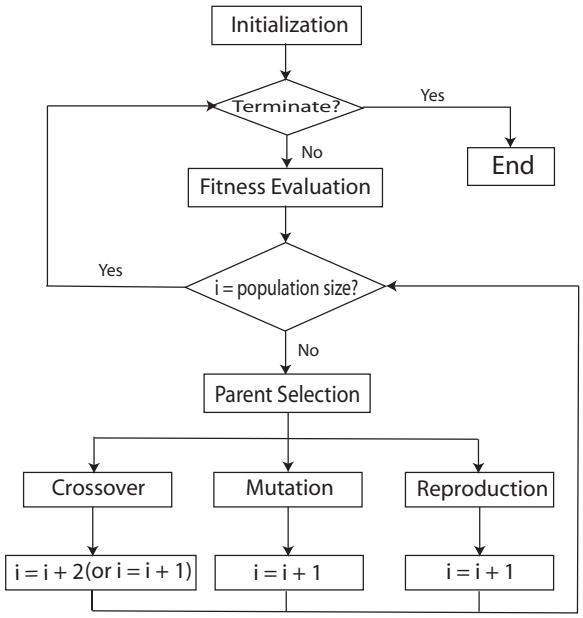


Figure 3.4: Genetic programming flowchart.

half method (one of the most common) is employed for initialization [34]. It utilizes the parameter D_{Max}, the maximum depth of the parse tree. The initial generation is generated by two methods with equal probability. 1) Full method: each branch of the tree has the same depth up to D_{Max}. 2) Grow method: the tree is constructed from a root with tree nodes chosen randomly from both the F and T sets as long as the depth of the tree does not exceed D_{Max}. The most common termination criteria either specify the maximum number of generations or the fitness threshold value. While the fitness evaluation function is specific to the particular problem, the process of variation can be abstracted from the problem domain. Crossover in GP is performed by swapping subtrees between two parents (in the most common case). Mutation is mostly implemented by replacing a subtree of the selected individual by a randomly generated subtree, with the constraint that the depth of the new individual does not exceed D_{Max}. This is known as single-point mutation. Reproduction simply copies the parent individual to create offspring.

When using GP to produce solutions, various combinations of selection and replacement mechanisms can be applied. For example, for parent selection, one can apply fitness proportional selection, where the probability of selecting an individual is a proportional to the fitness of that individual. After obtaining parent(s), GP applies genetic operations like crossover and mutation on the parent individuals and generates new individuals as offsprings. One then can specify how the offsprings replace individuals in the previous generation to construct a new generation. For example, one can use generational replacement to replace all individuals in the previous generation with newly generated offspring. After specifying the initialization procedure, the selection mechanism, the replacement scheme, the fitness evaluation process and related parameters, the GP method is executed, and solutions are evolved and evaluated.

3.5.2 Chemical Field Evolution via Genetic Programming

The chemical field function C around an individual MP has the form $1/\mathcal{C}(d, \theta, t)$. We define the secondary function $\mathcal{C}$ to be polynomial and trigonometric functions with variables of cell-cell distance d, one cell's angular position θ in another cell's local coordinate system, and simulation time t. We utilize a genetic programming framework to evolve the secondary function $\mathcal{C}$. We use $1/\mathcal{C}()$ in the aggregation simulation in order to increase the likelihood of creating a chemical field function that decreases with distance from an MP. In fact, since the variable d is a function included in the initial population, $1/d$ is a candidate solution evaluated in the first generation of the GP process. We also include real numbers E in the chemical field function. The chemical field functions are the genotypes of the evolutionary process and are stored as a prefix parse tree. The aggregated forms produced by the chemotaxis simulator are the phenotypes associated with the chemical field functions in a population.

The details of the morphogenetic primitive genetic programming process, as illustrated in Figure 3.2, are the following. The ramped half-and-half [34, 62] method is used to generate the initial population of functions. Each MP aggregation simulation employs one function from the population as the chemical field function for all of its MPs. A simulation is performed for each chemical field function, producing an image of the aggregated result that emerges from the local interactions directed by the function. Each aggregated result is then evaluated by the fitness function, which compares the aggregate image with a target image. A fitness value (a scalar between 0 and 1) is assigned to each individual that quantifies how closely the resulting aggregate matches the target shape. The fitness values are returned to the GP framework. Parent selection is then performed on the population based on the fitness value and variation takes place. Generational replacement is used for survival selection, which means that the offspring replaces the parents from the previous generation. The GP process repeats until the termination criteria are met, either after generating a maximum number of generations or the fitness of an individual surpasses the threshold, i.e., the associated aggregate is approximately the same shape as the target.

We can also seed the first generation with functions produced by a previous run of the shape composition process. For example, when trying to generate the hourglass shape (Figure 3.7 (left)), we first ran 50 generations of the GP-based shape composition process and then picked the 100 top individuals from all 50 of the generations to be the first generation for another cycle of a new GP process.

3.5.3 Fitness Function

The fitness function defines the similarity between two shapes, the aggregate generated by the MPs under the influence of the chemical field function and the user-defined target shape. Both the final aggregate from the MP simulation and the user-defined target shape

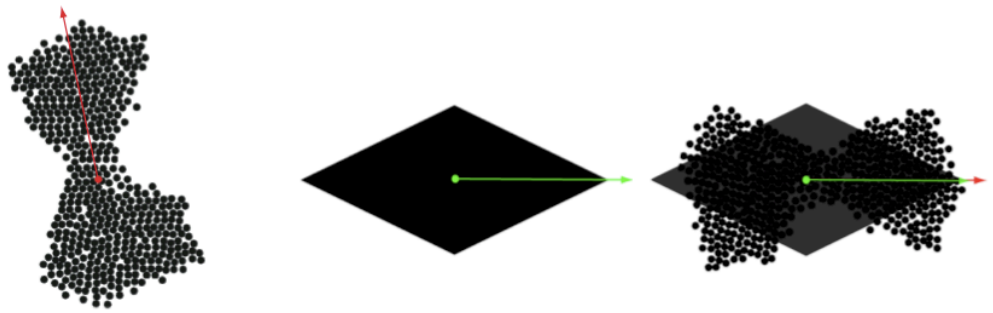


Figure 3.5: Aggregated MPs are aligned with a target shape. The square of the ratio of overlapping black pixels and the total number of black pixels in the target shape gives the fitness value for the aggregate.

are represented as black and white images. For the aggregates, the locations occupied by the MPs are marked with black pixels. The targets are represented with images of black solid geometric shapes.

The fitness function is based on comparing these images at the pixel level. It is important to keep the total area of the MPs (the area of one MP times the total number of MPs) approximately equal to the area of the target shape. Since MPs are not aware of a global coordinate system, we do not want the shape of the resulting aggregate to be tied to a specific fixed orientation. We therefore align the aggregated shape with the target shape before applying the fitness function. This is accomplished by applying a translation and rotation to the candidate aggregated shape that aligns its centroid and major axis with those of the target shape, as seen in Figure 3.5. This allows MPs to aggregate into the desired shape, but not necessarily in the same position and orientation of the target. Given the toroidal structure of the computational environment a specialized technique is required to calculate the centroid of the MP aggregate [3]. (See Appendix A for more details.)

Once the alignment is completed, the fitness function is calculated as the square of the ratio of the number of black pixels that overlap in the aggregate and target images and the

total number of black pixels in the target shape:

$$f = \left(\frac{\sum p_i, p_i \in S_a \cap S_t}{\sum p_j, p_j \in S_t} \right)^2, \tag{3.8}$$

where f is the fitness of the individual, p_i are the black pixels that are present in both the aggregate and target images, p_j are the black pixels in the target image, S_a and S_t are the aggregate and the target shape images. The floating-point number f, which has a value between 0 and 1, is assigned as the fitness of the chemical field function used to generate the aggregate shape.

A large number of computed aggregates have an amorphous, blob-like shape, which can produce significant overlap with most target shapes and a high fitness value. Squaring the ratio makes the fitness function more sensitive to shape similarity, and produces a high value only when the two shapes are truly similar. Therefore, adding the square term to the fitness evaluation function decreases the fitness value for these amorphous shapes, and thus makes it better at identifying aggregates that have the target's shape.

3.5.4 GP Parameters

The following summarizes the components in the GP process.

- **Individuals:** Each individual is represented as a prefix tree with a maximum depth $D_{Max} = 17$. Elements of the terminal set are stored in the leaf nodes and elements of the function set are stored in the internal nodes.

- **Function and terminal set:**

  $F = \{+, -, *, /, exp, log, sin, cos\}$ and $T = \{E, d, \theta, t\}$

  In order to maintain the closure property of the GP process [34, 62], we use protected division and protected logarithm, which allows for the most general description of

the functions. A division by zero is reprogrammed to be division by one. And the logarithm of a number smaller than 10^{-300} is clamped to be the logarithm of 10^{-300}, base e. Here, $E \in \Re$ is a real number; d is the distance between two MPs, θ is the angular location of one MP measured in the local coordinate system of another MP, and t is the simulation time.

- **Selection:** Parent selection is proportional to the fitness value and uses tournament selection (tournament selection size $k = 7$), which randomly chooses k individuals out of the population and returns the one with the best fitness. Generational replacement is used for survivor selection.
- **Variation:** When making new offspring a swap subtree crossover operation and single-point mutation, i.e., replacing a subtree with a stochastically generated subtree, are used.
- **Fitness evaluation:** We compare an image of the resulting aggregate with an image of the target shape. The comparison produces a fitness value, a floating-point number between 0 and 1, with 1 being the maximum fitness (similarity) value.

Parameters used to control the GP process are:

- **population size:** 100.
- **maximum number of generations:** 50 or 100.
- **crossover probability:** $P_c = 0.75$ *to* 0.9.
- **mutation probability:** $P_m = 0.1$ *to* 0.25.
- $\mathbf{D_{Max}}$: 17 (maximum parse tree depth),
- **k**: 7 (tournament selection size).

3.5.5 Distributed Master-Slave Model

Given the magnitude of the search space, i.e., all possible polynomial and trigonometric functions constructed from the function set and the terminal set in a parse tree of depth $\mathbf{D_{Max}}$, and the significant amount of computation needed to search and evaluate it, we have developed a distributed computing system to produce MP solutions [5]. Open Beagle [48], a C++ evolutionary computing framework, has been utilized and altered to function as the computational core of the distributed genetic programming environment. With Unix shell scripting [18], we implemented an n-slave, 1-master system for distributed computation of MP simulations.

The master process adaptively sends out individual functions over a network to a number of slave processes, and performs most of the genetic programming steps. Each slave node hosts one slave process. The slave process is responsible for incorporating the individual (chemical field function) into an MP simulation program, running the simulation, comparing the resulting aggregated shape with the user-defined shape and returning a fitness value to the master process. The master process then utilizes the fitness value associated with the individual function to perform genetic programming operations. It generates a new population, which is distributed once again to the slave nodes, and concludes the whole GP process once the termination criteria are met.

3.6 Results

3.6.1 MPs with Fixed Orientation

In the first experiments utilizing our approach, all MPs have the same, fixed orientation. MPs do not rotate and do not attempt to align with their neighbors, since they share a common coordinate system. The MP simulations performed during the GP process con-

sisted of 500 primitives, where each primitive has a radius of 5 units[1]. The MPs move in a toroidal computational environment of 500×500 units on a hexagonal grid, where locations on the grid are separated by 1 unit. Each simulation is run for 1000 time steps.

The target shapes given to the MP aggregation framework include an ellipse (Figure 3.6 (left)), a diamond (Figure 3.6 (right middle)), an hourglass (Figure 3.7 (left)), a cross (Figure 3.7 (right middle)), the letter 'b' (Figure 3.8 (left)), the letter 'S' (Figure 3.8 (right middle)), a star (Figure 3.9 (left)) and a triangle (Figure 3.9 (right middle)). The best resulting MP-generated shapes are placed next to the target shapes in these figures. The 'S' shape did not completely converge on the desired shape. While the target letter 'S' is a stand-alone object, the associated MP aggregate is a continuous shape that is connected along the top and bottom edge, because of the toroidal structure of the computational environment.

Since the field functions generated by the evolution process are very complex, we simplified them with Maple(TM) and conducted tests to ensure that the simplified functions still produce the same aggregate structures. The simplified field functions that produce these shapes are listed in Table 3.1, along with the number of generations needed to derive the original, unsimplified functions.

[1]One unit of length is equal to a pixel width.

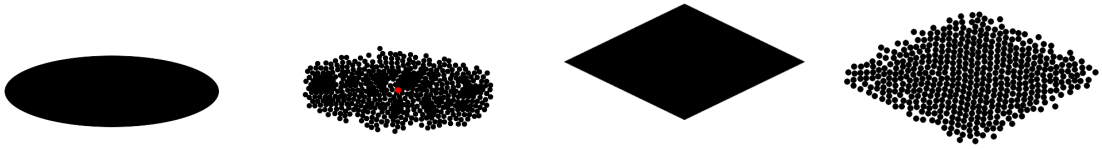


Figure 3.6: Ellipse: (left) Target shape. (left middle) Self-organized MPs. Diamond: (right middle) Target shape. (right) Self-organized MPs.

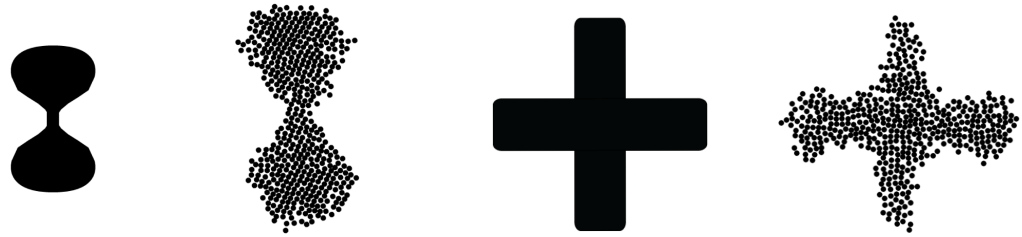


Figure 3.7: Hourglass: (left) Target shape. (left middle) Self-organized MPs. Cross: (right middle) Target shape. (right) Self-organized MPs.

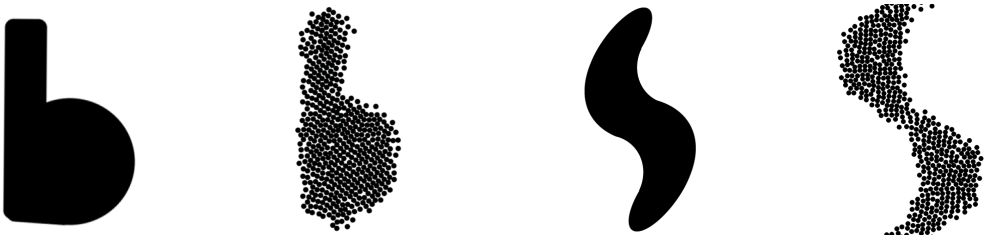


Figure 3.8: Letter 'b': (left) Target shape. (left middle) Self-organized MPs. Letter 'S': (right middle) Target shape. (right) Self-organized MPs.

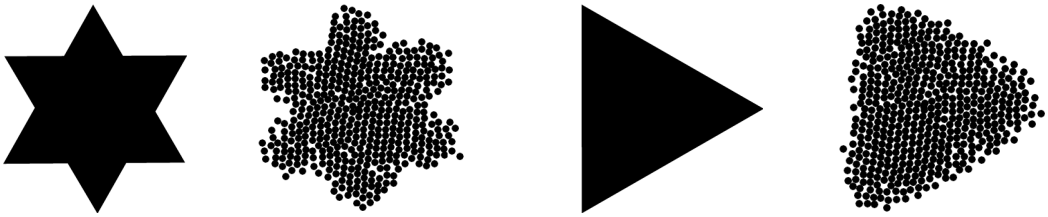


Figure 3.9: Star: (left) Target shape. (left middle) Self-organized MPs. Triangle: (right middle) Target shape. (right) Self-organized MPs.

Table 3.1: Summary of shapes, field functions, and the generation that produced the function.

Shape	Field Function	Generation
ellipse	$(\ln(d/t) - \ln(\cos(\theta)))\, t d^{-1}$	7th
diamond	$\left(\ln(d/t) - 0.924414\,\theta^{-1} - 0.36356\right) t d^{-1}$	11th
hourglass	$3.4950\,\theta^2 \cos\left(2d/\left(\ln\left(\cos\left(-\mathrm{e}^d + d\right)\right)\right)\right) t^{-1} + d$	54th
cross	$\ln\Big(2\,d - 2\,\theta + 0.94606\ (d - \cos(0.23102\,\theta)) \cdot \left(2\,d - 2\,\theta + 0.94606\,(\ln(d)\,(-2.4533 - 2\,\theta + d))/(\ln(\ln(d))) + 0.94606\,t^{-1}\right) \cdot (\cos(0.23102\,\theta))^{-1} + (\sin(t) + t)/(\cos(t)) + 0.94606\,t^{-1}\Big)$	30th
letter 'b'	$0.39755\,d + 0.39755\ \cos(\theta)\ + 0.39755\ \sin(\theta)$	14th
letter 'S'	$-\sin(0.67004 + \theta)\Big(d + 0.67004 + \Big(d - 2.1101\ (\sin(0.67004 + \theta))^2 - (d + 0.57590)\Big(-\sin(\sin(0.67004 + \theta))\left(-300\,(e^{0.0016497+t}\ln(10))/(t\,(t + d)) - t + \theta - 0.67004\right)^{-1} + 0.20668\,d\Big)^{-1}\Big)(\sin(0.67004 + \theta))^{-1} - (d + 0.57590)\Big(\sin(\ln(-0.52255\,\theta/t)\,(1.1275 + \cos(t)))\Big)^{-1}\Big)^{-1}$	27th
star	$\cos(\ln(\cos(\cos(\ln(t/d)))))$	42nd
triangle	$\cos(\cos(\ln(t/(\cos(\cos(\cos(\cos(\ln(d)))))))))$	47th

While these functions are complex compared to the simplicity of the target shapes, our results demonstrate the proof-of-concept that paradigms for spatial self-organization algorithms may be derived from the morphogenetic behaviors of cells. It has been shown that local interactions based on a chemotaxis paradigm can produce a self-organizing aggregation that leads agents to form into a user-specified shape. As stated in section 3.2, compared with other approaches, our method does not require the primitives to possess macroscopic shape information, to establish a global coordinate system, or to know their global positions in the environment. Nor does our method require a specialized seed primitive, or a global gradient information, which is generally used in potential field methods. In our method, user-defined shapes simply emerge from numerous local interactions among the self-organizing low-level primitives.

Given the stochastic nature of genetic programming, a number of pleasant surprises were produced during the evolution process. These are field functions that direct MPs to produce unwanted, but still quite interesting, shapes and patterns. A set of these surprising results are included in Figure 3.10. The most remarkable of these accidental shapes are the gear shapes in the first row.

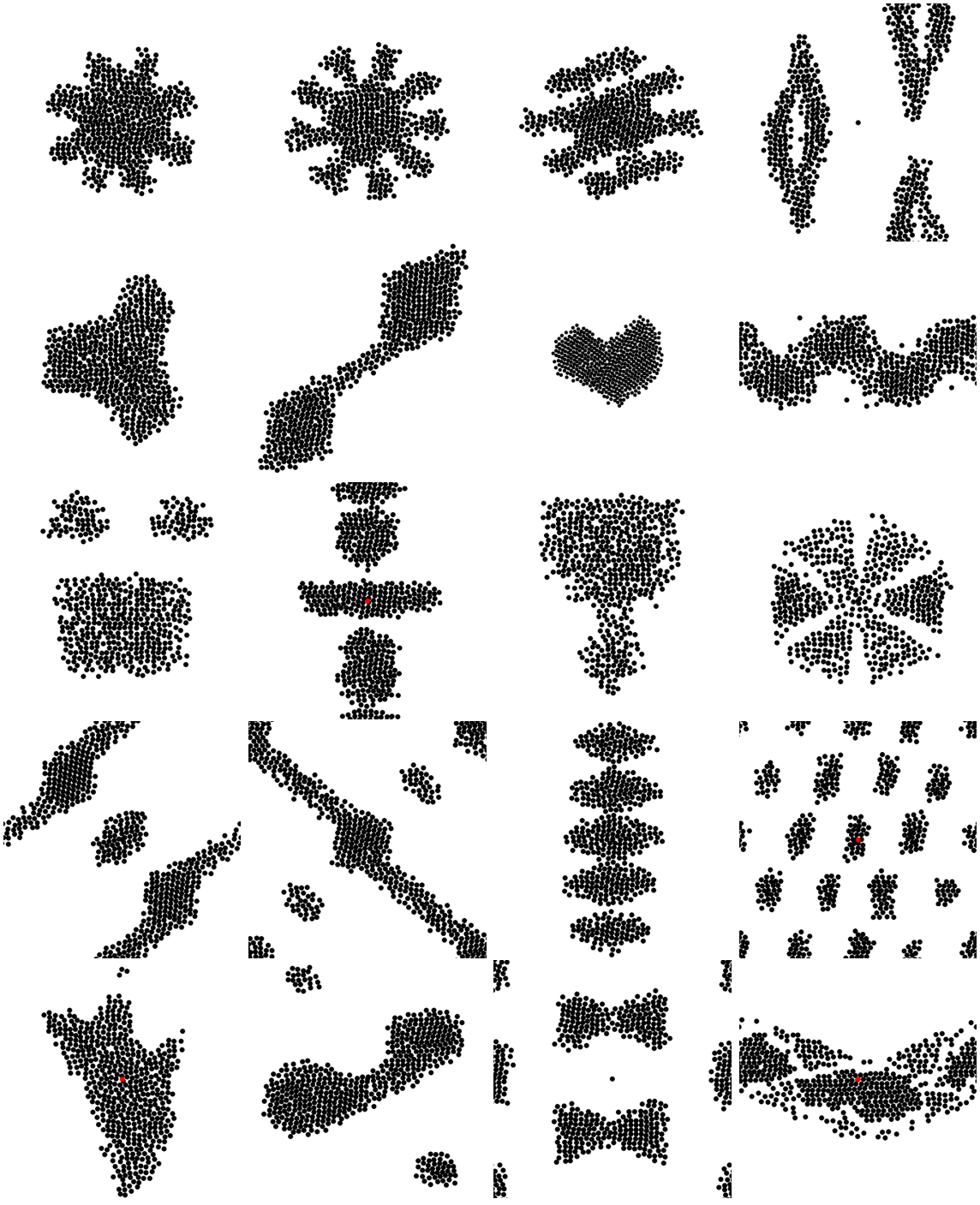


Figure 3.10: Unexpected shapes produced by a variety of MPs.

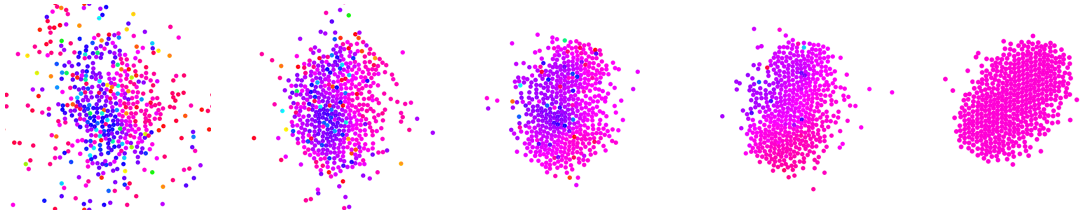
3.6.2 Rotating, Self-Aligning MPs

In the second set of experiments we investigated MPs' capacity to properly self-organize given that they are initialized with both random locations and orientations and allowed to align with the coordinate systems of their immediate neighbors. Our results demonstrate that MPs, given these random initial conditions, not only are able to form into user-specified shapes but can also self-align into a common orientation. The latter outcome empirically corroborates the theoretical result of Tanner et al. [98].

We used MPs with a common, fixed orientation in the genetic programming process when attempting to evolve chemical field functions for target shapes. This is the case for the shapes in presented in Figure 3.6 through 3.9. The field function evolution process with fixed orientations is computationally less expensive than with self-aligning MPs with arbitrary orientations (described in Section 3.4.4). We have found that the number of generations needed to generate field functions is shorter when MPs are not allowed to rotate and have a common coordinate system. However, the chemical field functions evolved with fixed orientation primitives are also able to direct primitives with different orientations to aggregate into the same target shape, as long as they self-align into a common orientation, as seen in Figures 3.11 through 3.18.

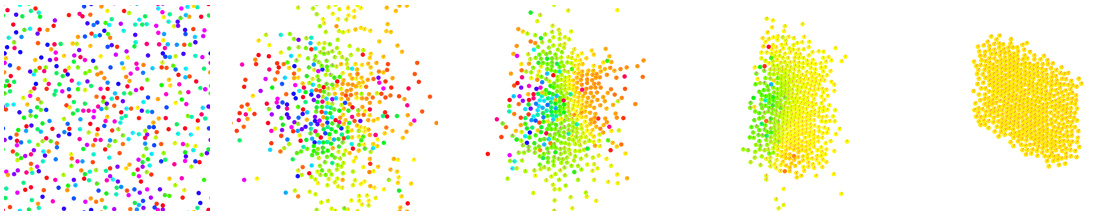
In these figures the orientation of an MP is visualized with the HSV color scheme by mapping orientation angle to the Hue channel. Here, $\omega = 0^\circ$ maps to red, $\omega = 120^\circ$ maps to green, and $\omega = 240^\circ$ maps to blue. Each MP is displayed as a colored disk, with a short line indicating its orientation. The horizontal right direction($+x$) is defined as $\omega = 0^\circ$. In the color examples it can be seen that the MPs both align to form a common coordinate system (the orientation colors become uniform) and self-organize into the desired shape, a typical result for our second set of experiments. The letter 'b' in Figure 3.15 is the one shape that

does not fully align all of its MPs. It should also be noted that the image sequences are not all taken at the same time intervals; thus explaining the varying degrees of mixing and self-alignment in the sequences.



(a) frame=240 **(b)** frame=280 **(c)** frame=320 **(d)** frame=360 **(e)** frame=1000

Figure 3.11: MPs self-aligning and self-organizing into an ellipse.

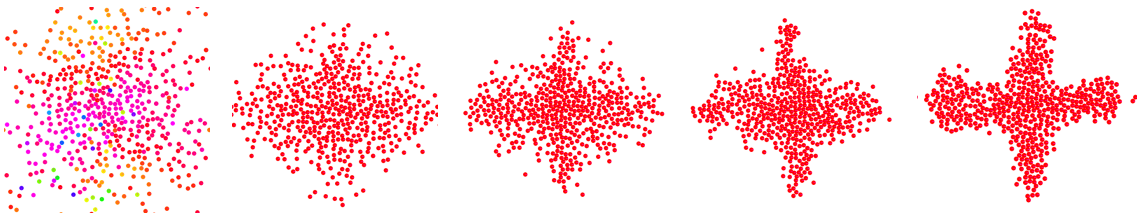


(a) frame=0 **(b)** frame=230 **(c)** frame=300 **(d)** frame=400 **(e)** frame=710

Figure 3.12: MPs self-aligning and self-organizing into a diamond.

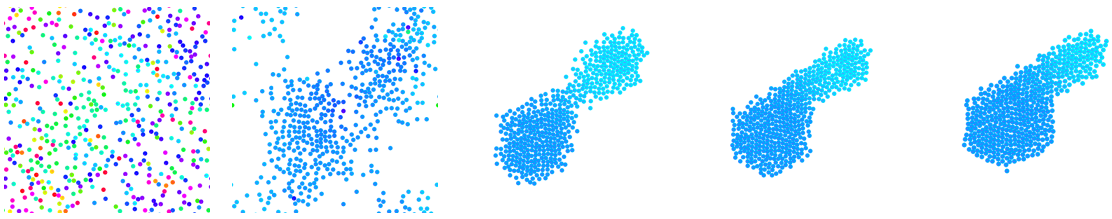
(a) frame=70 **(b)** frame=440 **(c)** frame=570 **(d)** frame=660 **(e)** frame=900

Figure 3.13: MPs self-aligning and self-organizing into an hourglass.



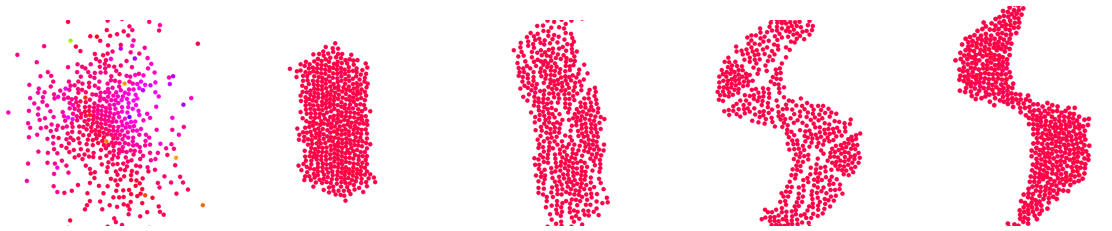
(a) frame=300 **(b)** frame=600 **(c)** frame=680 **(d)** frame=840 **(e)** frame=1000

Figure 3.14: MPs self-aligning and self-organizing into a cross.



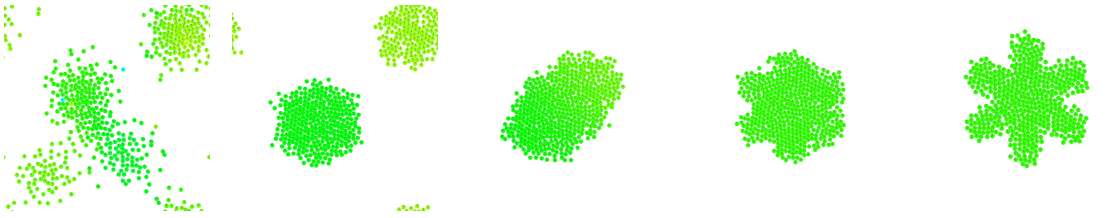
(a) frame=140 **(b)** frame=330 **(c)** frame=540 **(d)** frame=810 **(e)** frame=1000

Figure 3.15: MPs self-aligning and self-organizing into a 'b'.



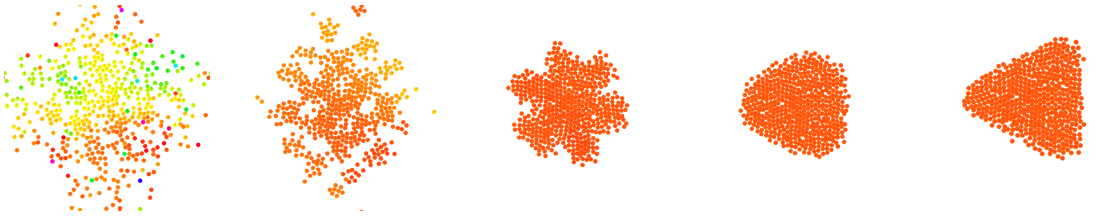
(a) frame=290 **(b)** frame=700 **(c)** frame=780 **(d)** frame=960 **(e)** frame=1000

Figure 3.16: MPs self-aligning and self-organizing into an 'S'.



(a) frame=350 **(b)** frame=620 **(c)** frame=700 **(d)** frame=970 **(e)** frame=1000

Figure 3.17: MPs self-aligning and self-organizing into a star.



(a) frame=420 **(b)** frame=580 **(c)** frame=720 **(d)** frame=900 **(e)** frame=1000

Figure 3.18: MPs self-aligning and self-organizing into a triangle.

3.6.3 Chemical Field Visualization

In order to better understanding the self-organization of the chemical fields of the MPs, we visualize the spatial distribution of the chemical concentrations. Visualizations of the chemical fields surrounding individual MPs are given in Figures 3.19, 3.20 and 3.21. Darker portions of the field represent regions of higher chemical concentration and the red regions are isoregions (regions where the chemical concentration have approximately the same value). Figure 3.19 presents the "static" chemical fields of the MPs that self-organize into the star, diamond, and letter 'b' shapes. While the star and diamond chemical functions do contain time variables, the changing value of time over the aggregation simulations does not significantly alter the form of these chemical fields. Figure 3.20 and 3.21 demonstrate how the individual chemical fields of the ellipse and letter 'S' MPs change as simulation time t increases.

The 'chemical' fields emitted by the individual MPs accumulate in the environment. It is this cumulative field that each MP senses and responds to by moving along its gradient. Figures 3.22 to 3.25 illustrate how the MP cumulative fields self-organize during the aggregation process. Red isoregions are drawn to further highlight the emerging structure of the fields.

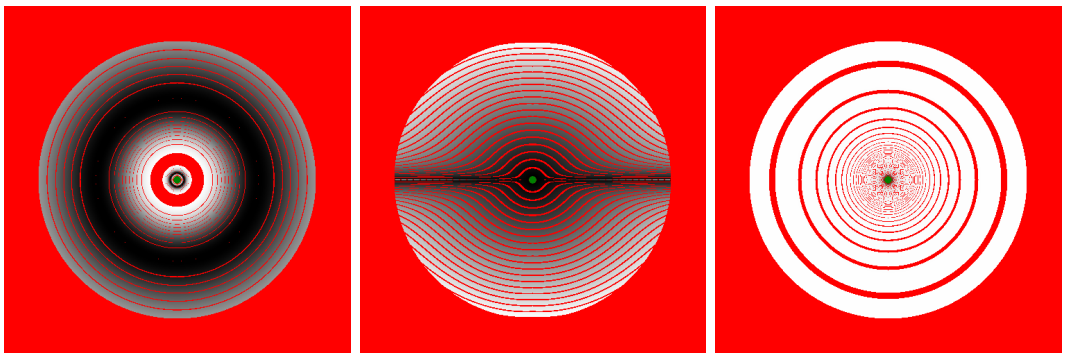


Figure 3.19: "Static" chemical fields emitted by the star, diamond and letter 'b' MPs.

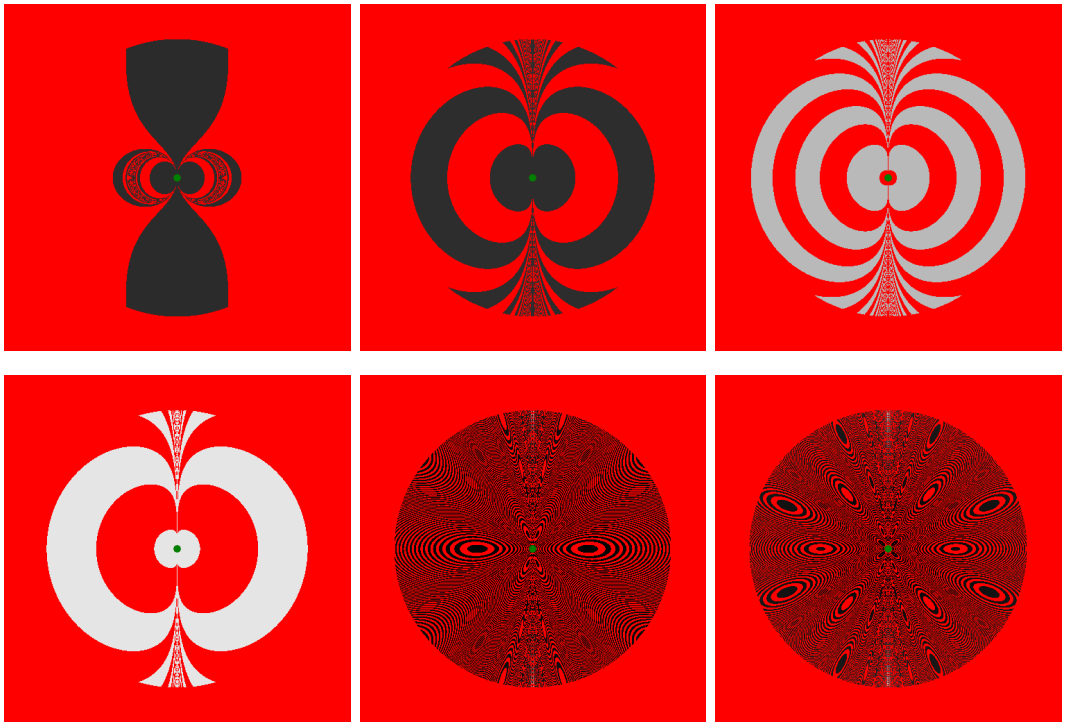


Figure 3.20: Dynamic chemical field emitted by the ellipse MP.

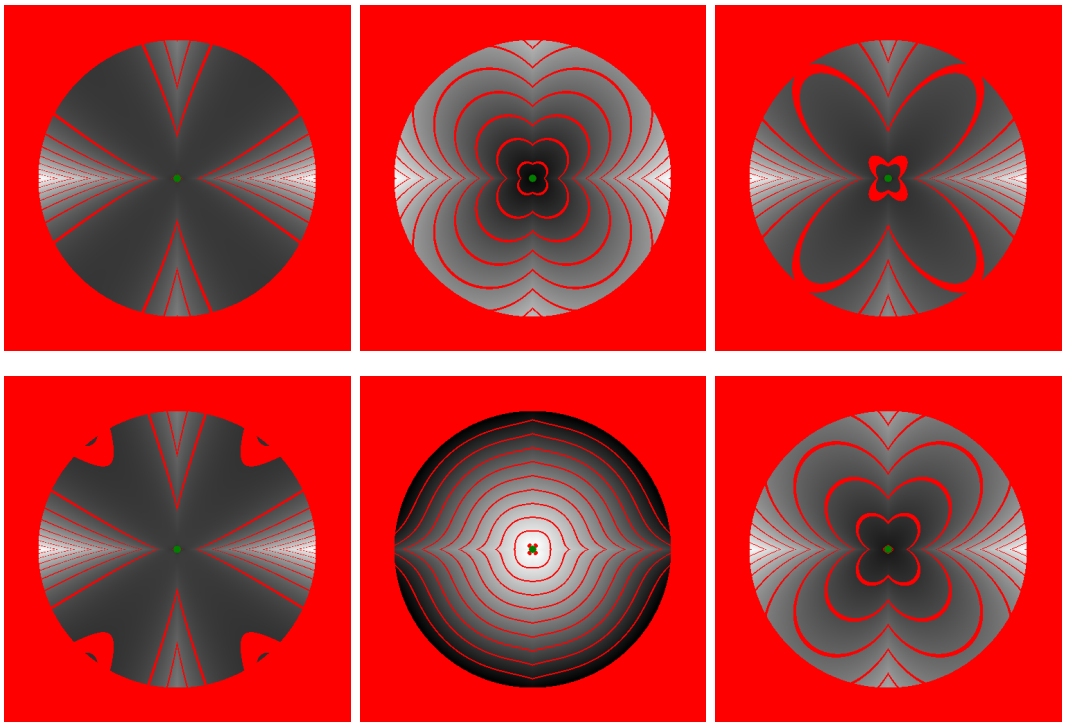
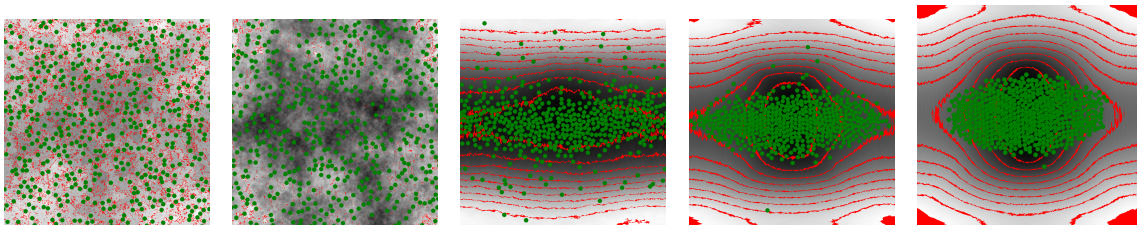
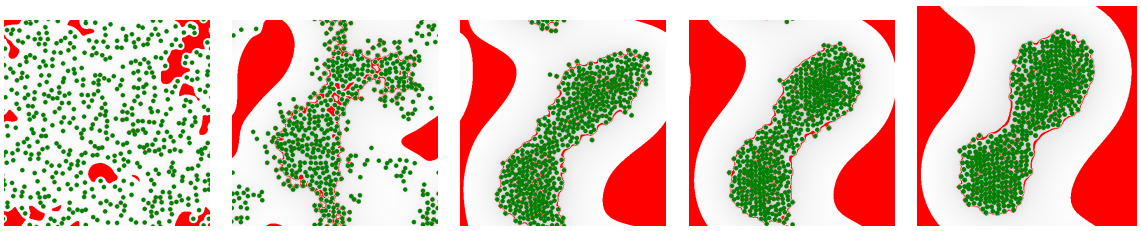


Figure 3.21: Dynamic chemical field emitted by the letter 'S' MP.



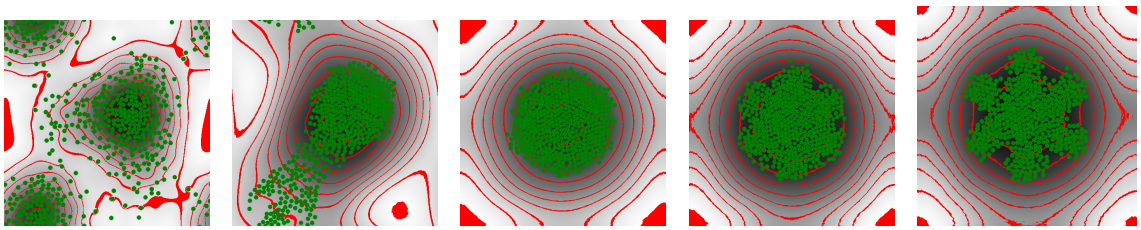
(a) frame=10 **(b)** frame=250 **(c)** frame=450 **(d)** frame=570 **(e)** frame=960

Figure 3.22: Cumulative chemical field emitted by a set of MPs self-organizing into an ellipse shape.



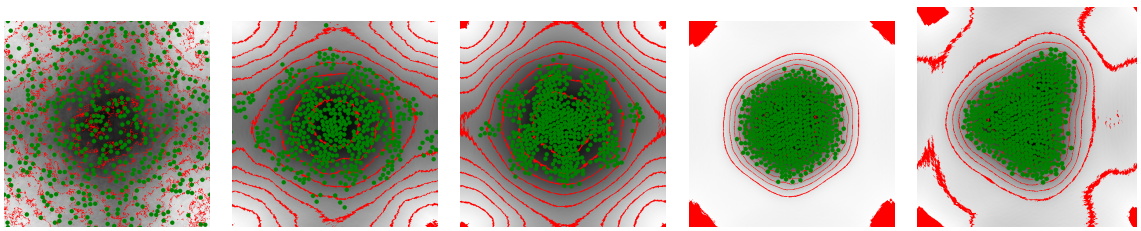
(a) frame=90 **(b)** frame=400 **(c)** frame=560 **(d)** frame=740 **(e)** frame=1000

Figure 3.23: Cumulative chemical field emitted by a set of MPs self-organizing into an hourglass shape.



(a) frame=310 **(b)** frame=690 **(c)** frame=840 **(d)** frame=970 **(e)** frame=1000

Figure 3.24: Cumulative chemical field emitted by a set of MPs self-organizing into a star shape.



(a) frame=370 **(b)** frame=520 **(c)** frame=690 **(d)** frame=840 **(e)** frame=970

Figure 3.25: Cumulative chemical field emitted by a set of MPs self-organizing into a triangle.

3.7 Discussion

3.7.1 Generating Results

While the general procedure for discovering the chemical fields has been outlined in previous sections, in practice the process of generating the field function for a specific shape can be quite arduous. At this point in our work we are still discovering which shapes our approach is capable of producing. In the process of attempting to discover field functions, parameters for the approach itself were varied to find the ones that produced the desired result. Those parameters include the probability of following the chemical gradient, the magnitude of random movements, and the balance of mutation versus cross-over in the GP stage. Given the exploratory nature of our work, we have experienced failures when trying to make different target shapes. For example we failed to find field functions that produced a heart shape and the letters O, T, U and Y.

Many times the GP process does fail to generate a proper target shape and is caught in an undesirable local minimum configuration instead. For example, when attempting to generate the letter 'b' shape, frequently the MPs form into a single blob or an hourglass shape. One way to solve this problem is to introduce more randomness (or noise) into the system. For the GP process we increased the probability of mutation, and for the MP aggregate simulation we randomized the order of cell movement processing. The modifications to the GP process introduced new equation segments into the individuals, i.e the chemical field functions. Making the order in which cells are moved during the processing of a single simulation time step increased the likelihood that a function would consistently produce a particular shape. In other words, adding randomness to the aggregation simulations made them more stable and predictable. This correlates with our previous cell sorting research that showed that noise is necessary for producing highly ordered, layered cell structures

[37]. During the development of our approach we also found that we needed to improve the sensitivity and accuracy of our fitness function, by squaring the ratio in Equation 3.8.

3.7.2 Robustness

During the aggregation process, each MP starts with a random position and possibly a random orientation. In order to evaluate the reliability/robustness of our approach, 100 aggregation simulations were performed for the ellipse, diamond, hourglass, cross, letter 'b', letter 'S', star and triangle shapes, each with different initial conditions. One set of experiments uses MPs with fixed orientations but uniform random initial positions. The other set of experiments uses MPs that self-align additionally with initial random orientations. The outputs of these simulations were visually evaluated to compute a robustness statistic, the percentage of the aggregation results that produces each desired final shape. The robustness information is summarized in Table 3.2. It can be seen that only a few shapes, e.g. ellipse, diamond, cross, letter 'S' and the star, can be produced consistently, with the ellipse and cross MPs being robust both for the fixed orientation and self-aligning MPs. It is interesting to note that the robustness of the shape formation process significantly decreases for some shapes when including self-alignment, e.g. for the ellipse, diamond and letter 'S'. However for some shapes initialization with random orientation followed by local alignment improves the robustness, e.g. for the star and triangle. Overall, these statistics highlight the fact that self-organizing MPs do not always aggregate into the desired form. Work described in later chapters present our efforts to control the shape formation process, and therefore improve its robustness and reliability.

Table 3.2: Robustness of the aggregation simulations with fixed orientation MPs and self-aligning MPs. Listed are the percentage of simulations out of 100 runs that produced a desired shape for a variety of MPs.

Shapes	ellipse	diamond	hourglass	cross
Fixed-orientation MP	100%	99%	10%	100%
Self-aligning MP	87%	27%	4%	100%
Shapes	letter 'b'	letter 'S'	star	triangle
Fixed-orientation MP	9%	99%	50%	7%
Self-aligning MP	7%	18%	84%	39%

3.7.3 Dynamic Stability

Since simulation time t is included as a variable in the chemical field function, our approach effectively evolves field functions that direct MPs to aggregate into the desired shape by a certain time (1000 iterations in our experiments). We conducted tests to determine whether the resulting aggregates are stable, unstable or periodic structures, i.e. do MPs reach their desired shape and maintain it, do MPs attain their desired shape by 1000 iterations then collapse into a another shape, or do they oscillate between two shapes. In our tests we extended the total number of iterations to 1500. After iteration 1000, t was held constant; thus fixing the structure of the individual chemical fields for the remainder of the simulation. Since the probability of an MP following the chemical concentration gradient is a function of simulation time as well, as in Equation 3.7, the probability of following the gradient was set to 1.0 once the iterations exceeded 1000. MPs always followed the chemical field gradient in the later stages of these tests.

The results of these tests for MPs that achieved their target shape at 1000 iterations are summarized as the following. The ellipse, diamond and star aggregates always maintain their target shape. The hourglass, letter 'b' and letter 'S' shapes keep their target shapes approximately 50% of the time. The cross aggregates turn into a hexagonal form and the

triangle aggregates turn into a bow tie shape. These experiments suggest that the self-organizing systems that form the ellipse, diamond and star shapes are stable after time step 1000; while the hourglass, letter 'b' and letter 'S' MPs are semi-stable, while the cross and triangle are unstable at least in terms of producing the desired shapes. Exploring and understanding this complex dynamic behavior will be the focus of future work.

3.7.4 Running Time

A significant amount of computation is needed for the genetic process to produce the chemical field function for a particular shape. Each 1000-time-step MP simulation requires approximately 4-CPU to 30-CPU minutes on a 2.4 GHz Opteron processor, depending on the complexity of the chemical field function. Initially the GP calculations were performed on an 8-node cluster, and then later were migrated to a 64-core cluster. Since the overhead of distributing the computation over the nodes is quite small compared to the simulation times themselves, we found that we achieved nearly linear speed-ups in our distributed GP system [5], with an approximately 64× speed-up on the 64-core cluster.

Given our computing cluster and the time required for each aggregation simulation, each 100-individual generation requires approximately one hour of actual time to compute. Since the GP process may produce the desired function on average by the 30th generation, a successful run can derive a shape-specific field function in one to two days. In our experience though, several of these day-long runs are needed to find the correct parameters to produce the desired result. Of course, once we have the correct chemical field function only a small amount of computation (approximately 15 CPU-minutes) is needed to simulate the desired aggregation behavior.

3.8 Conclusion

We have presented a new approach to shape formation using self-organizing primitives whose behaviors are based on cell biology. The key concept is that local interactions between the 2D primitives, called morphogenetic primitives (MPs), direct them to aggregate into a user-defined macroscopic shape. The interactions are based on the chemotaxis-inspired aggregation movements exhibited by actual living cells. Here, cells emit a chemical into their environment. Each cell responds by moving in the direction of the cumulative chemical field gradient detected at its surface. MPs, though, do not completely mimic the behavior of real cells. The chemical fields are explicitly defined as mathematical functions and are derived via genetic programming. A fitness measure, based on the shape that emerges from the aggregation process, directs the evolution of the function. We have implemented a distributed GP system to speed up the evolution process. We have successfully applied this method to evolve the interaction rules to form a number of simple shapes and have conducted preliminary analysis of the overall approach. Our general shape formation primitives are truly self-organizing, compared with previous works [72, 73, 108, 28]. The primitives do not know their global position or contain a blueprint of the final shape they are going to form. The primitives are not assigned a location in the final configuration. The primitives only interact with neighboring agents, basing their actions on local and internal information. Compared to previous work, our system has achieved a deeper level of self-organization.

While we are able to discover local interaction rules for a number of user-predefined shapes, we learned from our repeatability tests that our system can not always consistently produce the specified outcome. We aim to discover methods for making the self-organizing shape formation robust. In Chapter 2, we are able to achieve a consistent sorted structure by identifying the proper range for system parameters. Here we want to achieve robustness

by not only changing parameter values of our system, by also studying and controlling the shape evolution process of the primitives. For analysis purposes, we simplify the system by eliminating the toroidal environment. We extend the size of the arena and initialize primitives in a round disc at the center. The specific goal is to discover the properties that consistently lead to a desired self-organizing shape formation. Furthermore, we can utilize these properties to create certain simulation conditions that robustly produce the aggregation of the desired shape.

Chapter 4: Predicting Spatial Self-Organization

4.1 Introduction

We have developed a self-organization shape formation system for generating user-specified macroscopic shapes and structures (Figure 4.1 and Chapter 3), based on chemotaxis-based local interactions [10]. In some cases, we have observed that the morphogenetic primitives (MP) do not spatially self-organize into a unique shape, but instead form two or more stable final configurations. We aim to find a way to characterize the macroscopic behavior of the MP swarm. Towards this end, we have discovered features, based on statistical moments and their derivatives of the agents' positions that can distinguish different final configurations and that can be computed early in the swarm's evolution to predict the final shape of the self-organizing system. Analyzing these features with a Support Vector Machine [24], a supervised machine learning technique, classifies the agent system, i.e. provides a prediction for the system's final configuration. The accurate prediction based on statistical moments demonstrates that these features indeed capture the macroscopic structure of the MP swarm, a capability that we will harness in the next chapter to control the swarm.

4.2 Related Work

Multi-agent approaches have been widely utilized to model self-organizing systems [78, 87, 104]. Our current work focuses on how to analyze and predict the outcome of such systems. To date, a variety of methods have been used to analyze these systems. These methods usually require identifying global variables [60] and/or deriving quantitative properties of the whole system [97]. Differential equations also are used to capture state transitions of

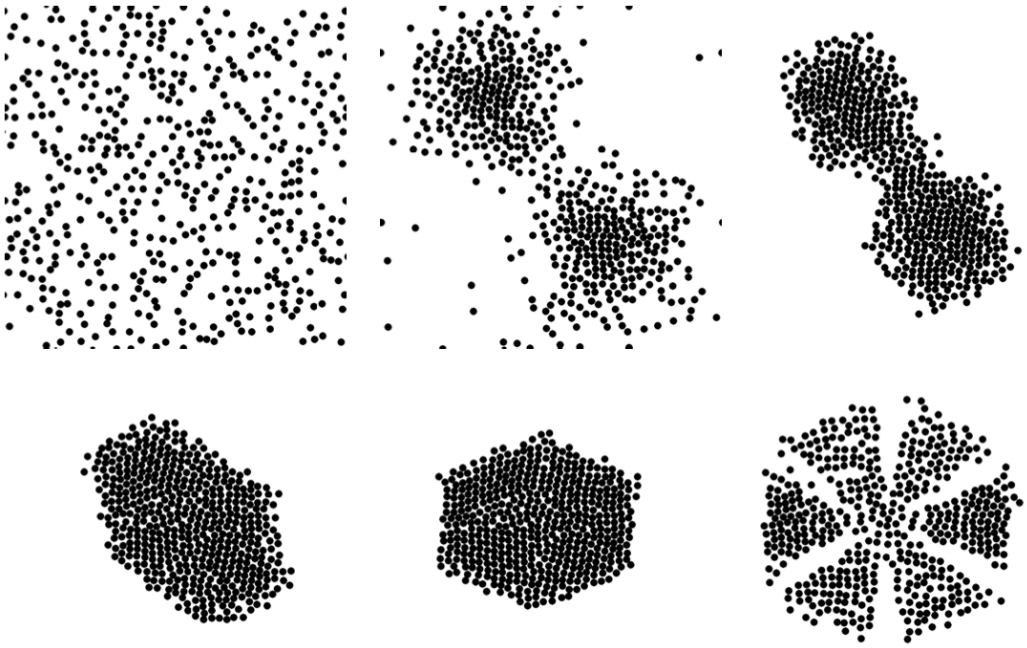


Figure 4.1: MPs self-organizing into a pinwheel shape.

a system [21, 42]. These methods require identifying global variables and the relationships between them, which can be obscure and quite complex.

Various analytical methods have been used to predict the outcomes of self-organizing systems. Lerman et al. showed that even though the behavior of an individual robot is stochastic, the collective behavior of a robot swarm is often predictable with a simple probability description [65].Time series analysis has been applied to predict self-organizing maps [89]. The analysis aims to predict long-term trends, rather than predicting specific final states. Hamann presents a data-driven, iterative process, based on a swarm urn model, Markov chains and abductive reasoning, for determining, testing, and refining hypotheses about how self-organizing decision-making system operate [53]. The process is able to identify influential feedback loops and important subsections of the configuration space in these types of systems.

Machine learning algorithms have also been applied to self-organizing systems. An unsupervised learning algorithm is utilized to construct a self-organizing map for information retrieval [66]. Support Vector Machines have been used together with self-organizing maps for shape-based image retrieval [111].

Statistical moments are measures that quantify the shape of distributions. In statistics, the method of moments is used to estimate population parameters [112]. In our work, we gain insights from the moments of the coordinates of moving agents in a $2D$ arena and predict the final outcome of the system based on the moment information.

4.3 System Modifications

To more easily analyze the self-organizing shape formation process, we first need to simplify the above system. The original local field function is hard to analyze because time t is also a variable in the function, which significantly increases the complexity of the system. Removing time t from the field function should lead to MPs that have long-term convergent, and stable configurations. In previous experiments, we have observed that interesting shapes can be obtained by field functions that are only functions of d and θ or even only d. Therefore, we simplified the local function to be $f = f(d, \theta)$ for our current experiments in order to facilitate the study of the self-organization process of primitive aggregation.

We also modified the arena that shape primitives move in. We eliminated the toroidal environment used in previous experiments and now prevent cells from moving across the arena boundary. We initialize the primitives inside a circle in the middle of the arena, with a diameter that is the same size as the width/ height of the previous environment.

4.4 Moment Analysis

In order to predict the final outcome of a self-organizing shape formation simulation, we first extract features that capture the spatial distribution of the MPs. Moments provide a quantitative way to describe a distribution. Since MPs are defined as small discs, we use the center of each disc to represent each MP's location. We therefore can simplify the collection of MP locations as a set of $2D$ points, and apply moment analysis to this set over the duration of the MP simulation.

4.4.1 Moment Calculation

We calculate the mean (first moment), variance (second central moment), skewness (third central moment) and kurtosis (fourth central moment) from the x and y coordinates of the MP centers. We analyze the locations X_i of all points (MPs) as a whole, rather than tracking the location and movement of each individual point. The population size of the agents is denoted as n, ($n = 500$), and the formulas of the four moments M_1 to M_4 are given in Equations 4.1 to 4.4,

$$M_1 = \frac{1}{n}\sum_{i=1}^{n} X_i, \tag{4.1}$$

$$M_2 = \frac{1}{n}\sum_{i=1}^{n} (X_i - M_1)^2, \tag{4.2}$$

$$M_3 = [\frac{1}{n}\sum_{i=1}^{n} (X_i - M_1)^3]/(M_2)^{3/2}, \tag{4.3}$$

$$M_4 = [\frac{1}{n}\sum_{i=1}^{n} (X_i - M_1)^4]/(M_2)^2. \tag{4.4}$$

These statistical moments provide quantitative information about the shape of histograms/ distributions. When computed for the x and y coordinates of the MPs, these moments capture the symmetry and asymmetry of the spatial distribution of the whole

population. Since the x and y coordinates of the points change over time, so do the four moments of the distribution of x and y values. The change of the moments as a function of simulation time also provides insight into the dynamic nature and structure of a particular MP simulation.

We define the time derivative of the moments as the slope of a linear interpolating function of consecutive moment values. A series of values are obtained by sampling the simulation time with a fixed interval τ. At each simulation time t, the four moments $M_i(t)$ $(i = 1\ to\ 4)$ of the overall distribution are calculated. For each moment M_i, we consider an interval of 4τ which containing 5 samples, that is, $M_i(t-2\tau)$, $M_i(t-\tau)$, $M_i(t)$, $M_i(t+\tau)$ and $M_i(t+2\tau)$. A straight line is fit to the five points with a least squares method. For each moment M_i at time t, we find the k_i and b_i that minimize Equation 4.5,

$$min \sum_{j=-2}^{2} |k_i \cdot (t + j\tau) + b_i - M_i(t + j\tau)|^2. \tag{4.5}$$

The slope k_i of the fitted line is then used as an approximation of the time derivative of $M_i(t)(i = 1\ to\ 4)$.

By calculating the moments and their time derivatives for both the x and y coordinates of the point set, at a given time t, we obtain a 16-dimensional vector to represent the distribution,

$$[M_{x_1}(t), M_{y_1}(t), M_{x_2}(t), M_{y_2}(t),$$

$$M_{x_3}(t), M_{y_3}(t), M_{x_4}(t), M_{y_4}(t),$$

$$k_{x_1}(t), k_{y_1}(t), k_{x_2}(t), k_{y_2}(t),$$

$$k_{x_3}(t), k_{y_3}(t), k_{x_4}(t), k_{y_4}(t)]$$

Our simulations consist of ($n = 500$) $2D$ points and therefore have a configuration space of dimension $2n$. By distilling an MP spatial distribution down to this feature vector, we reduce the dimensionality of the simulation from 1000 to 16.

Given the sensitivity of non-linear dynamical systems to initial conditions [109], it makes it extremely difficult, if not impossible, to predict the outcome of our complex, self-organizing system from its initial, random spatial configuration. We therefore attempt to predict the final spatial configuration at an early stage of the aggregation, usually before it is visually evident what shape will emerge from the process. We have found that applying our analysis at a time in the simulation that is a small percentage of the total time needed for the final aggregated shape to form (e.g. 5% to 10%) produces acceptable results (81% to 91% accuracy).

4.4.2 Using Support Vector Machines

We consider prediction of the bifurcating outcomes as a classification problem and utilize support vector machines (SVMs) [24] to solve it. An SVM applies kernel methods to labeled training data and transforms the data into a high dimensional space. It computes the optimal separating hyperplane, which is the one producing the largest margin between the two classes. Doing so produces a trained SVM model with kernel and constraint parameters. Classification is performed by mapping new data into the same high dimensional space and determining on which side of the hyperplane the new data lies.

Since our data is not linearly separable, we apply a Radial Basis Function (RBF) kernel in our SVM. We use LIBSVM [22] to perform SVM training and testing. To train the SVM, we employ a leave-one-out cross-validation method. We use grid search to find the best cost parameter C for the SVM and γ parameter for the RBF kernel. Once the optimal C and γ values were identified, accuracy statistics were gathered from numerous simulations, each

Table 4.1: Dataset information and prediction accuracies for bifurcating shapes at 10% (5% for ellipse) of simulation time.

Quarter-moon				
Total Time	Prediction Time	Positive Instances	Negative Instances	Kernel
35,000	3,500	100	100	RBF
Cost (C)	gamma (γ)	Accuracy	Sensitivity	Specificity
512	0.03125	91.5%	92.0%	91.0%

Ellipse				
Total Time	Prediction Time	Positive Instances	Negative Instances	Kernel
10,000	500	100	100	RBF
Cost (C)	gamma (γ)	Accuracy	Sensitivity	Specificity
2048	4.883e-04	89.0%	89.0%	89.0%

Four Discs				
Total Time	Prediction Time	Positive Instances	Negative Instances	Kernel
15,000	1,500	100	100	RBF
Cost (C)	gamma (γ)	Accuracy	Sensitivity	Specificity
512	1.221e-04	83.0%	84.0%	82.0%

Parallel Line Segments				
Total Time	Prediction Time	Positive Instances	Negative Instances	Kernel
10,000	1,000	508	100	RBF
Cost (C)	gamma (γ)	Accuracy	Sensitivity	Specificity
4	0.125	$(80.9 \pm 2.5)\%$	$(80.4 \pm 3.5)\%$	$(81.4 \pm 2.6)\%$

utilizing different random initial conditions.

To obtain training data for a specific target shape, we run a certain number (200) of MP simulations starting with different, random spatial configurations. Each simulation consists of a fixed number of time steps ($10,000$ to $35,000$), depending on how long it takes for the target shape to form. We save the spatial distribution of the agents every 100 steps of the simulation, that is, $\tau = 100$. Each simulation will produce an aggregated structure at the end of the simulation, and we label all final structures with one of two categories.

4.5 Results

We have applied the proposed prediction method to a number of bifurcating shape evolutions. The target shapes are a quarter-moon, an ellipse, a set of four discs, and a set of two parallel line segments. Once the SVM parameters with the best overall accuracy are determined through grid search and leave-one-out cross-validation, we calculate overall accuracy, sensitivity and specificity for each example. These results are summarized, along with the dataset descriptions and SVM parameters, in Table 4.1. The quarter-moon, ellipse and four-discs datasets each contains 200 simulations, with 100 belonging to each class. The parallel line segments dataset is unbalanced with a ratio of approximately $5:1$, with 508 instances belonging to one class and 100 belonging to the other.

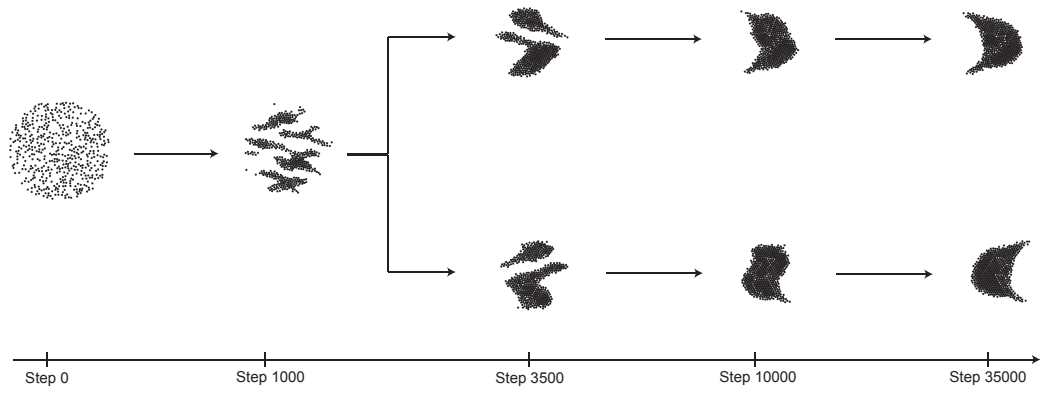


Figure 4.2: Shape evolution of the quarter-moon.

The quarter-moon dataset contains 100 left-facing and 100 right-facing structures. A typical shape evolution for this dataset is shown in Figure 4.2. The simulation reaches a stable state by $35,000$ simulation steps. We extract feature vectors in the form of Vector 4.6. At step $3,500$ (10% of the total simulation time) we are able to predict the left-facing or right-facing outcome with an overall accuracy of 91.5%.

The ellipse dataset contains 100 single ellipses and 100 non-single ellipses. Figure 4.3

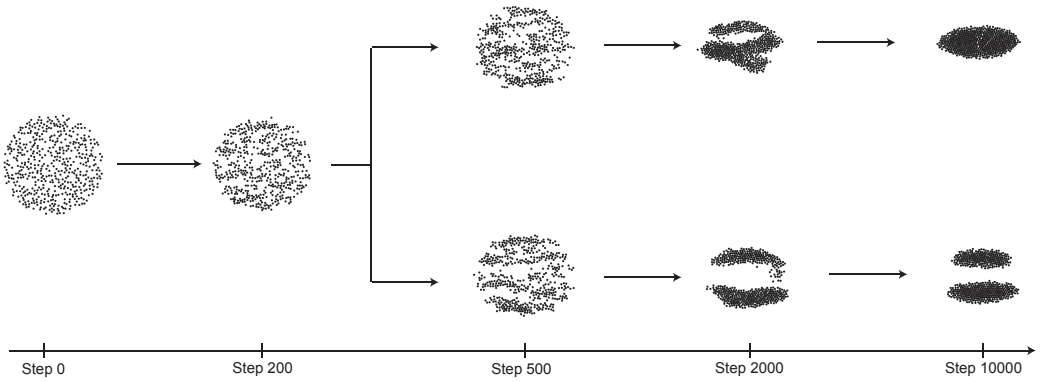


Figure 4.3: Shape evolution of the ellipse.

demonstrates its typical shape evolution. The simulation reaches a stable state by $10,000$ steps. At 500 steps (5% of simulation time) we are able to predict whether the simulation will produce a single ellipse or not with an overall accuracy of 89%.

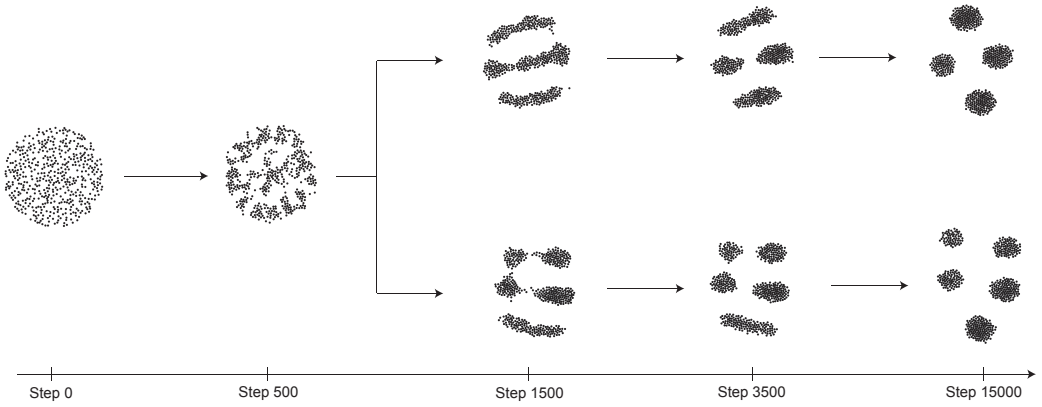


Figure 4.4: Shape evolution of the discs structure.

The four-discs dataset contains 100 four discs structures and 100 structures of five discs, with a typical shape evolution shown in Figure 4.4. The simulation reaches a stable state by $15,000$ steps. At $1,500$ steps (10% of simulation time) we are able to predict whether the simulation will form four discs or not with an overall accuracy of 83%.

The parallel line dataset contains 508 instances of two vertical parallel line segments

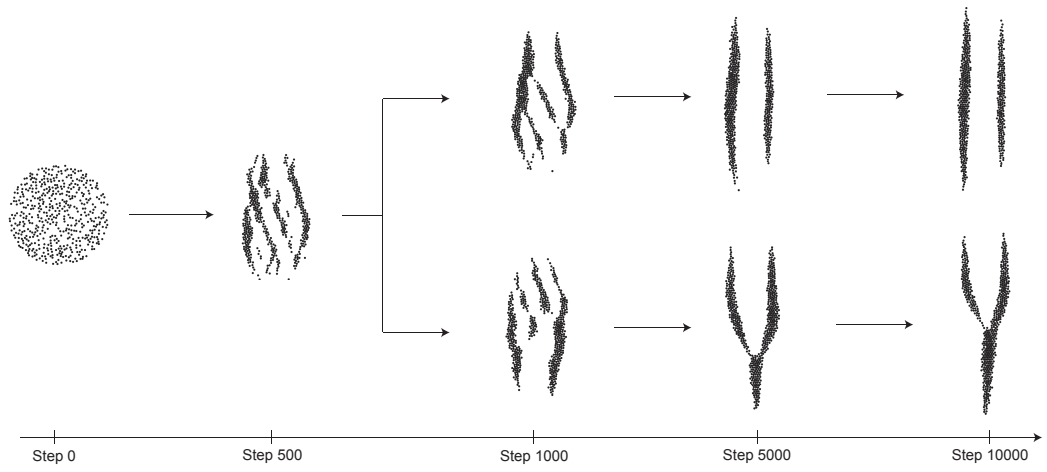


Figure 4.5: Shape evolution of the line segments structure.

and 100 instances of either "Y-shape" structures or one vertical line, as seen in Figure 4.5. The simulation reaches a stable state by $10,000$ steps. We calculate the prediction of the final configuration at step $1,000$. Since this is an unbalanced dataset, we under-sample the majority set. We randomly choose 100 out of the 508 two-line-segments instances and merge them with the minority set. Leave-one-out cross validation is then performed on the newly constructed balanced dataset. We achieve an average overall accuracy of $80.9 \pm 2.5\%$ over 100 randomly constructed balanced datasets when predicting the formation of two vertical lines.

Since the overall line-segment results are produced via averaging the results from numerous randomly generated partitions, we include a standard deviation with the average. The other examples have balanced datasets, and therefore leave-one-out cross-validation produces a unique partitioning and no deviations.

4.6 Conclusion

Based on the statistical moments of the agents' positions and the time derivatives of these moments, we are able to predict the final outcome of a spatial self-organization process at 5% or 10% percent of the simulation time with an overall accuracy of 81% to 91%. The accurate prediction based on statistical moments demonstrates that the moments successfully capture the macroscopic dynamic structure of the MP swarm. In the next chapter we show that these descriptive macroscopic features can be used to control or influence the agents' interaction so that they consistently form into one stable configuration, instead of bifurcating into two or more final configurations.

In this chapter, we use a global calculation of the statistical moments to analyze the agent swarm. Further work will explore methods for calculating statistical moments at the local level. This may require building a unified coordinate system via consensus.

Chapter 5: Directing Spatial Self-Organization

5.1 Introduction

We have observed in our previously developed chemotaxis-based self-organizing shape formation system (Figure 5.1 and Chapter 3), the morphogenetic primitives (MP) do not always spatially self-organize into a unique shape, but instead may form two or more stable final configurations. Given this sometimes inconsistent behavior, it would be useful to control the outcome of these bifurcating spatial self-organization processes. This would allow us to guarantee that all of our MP simulations produce a single, desired shape. In Chapter 4, we analyzed whole swarm populations at a global level in search of macroscopic, distinguishing attributes. This analysis identified features, based on statistical moments of the agents' positions, that have significantly different values for different outcomes of a swarm aggregation. We also discovered that these statistical moments can be used to accurately predict the outcome of the self-organization process at an early stage of the shape aggregation [9]. Given these differentiating moments, we aim to employ them to direct the outcome of our self-organizing system via generating biased, random initial conditions.

Through our study of the dynamics of a swarm's aggregation process we noted the connection between initial conditions and the final shape configuration of the swarm. See Chapter 2.5.3. Given our experience with changing the initial conditions of the cell sorting simulations and the predictive capabilities of an MP swarm's statistical moments, we hypothesized that biased, random initial conditions that meet specified constraints, i.e. have well-defined statistical properties, could consistently yield simulations with a unique final outcome. For those agent interactions that ultimately produce bifurcating/multiple final

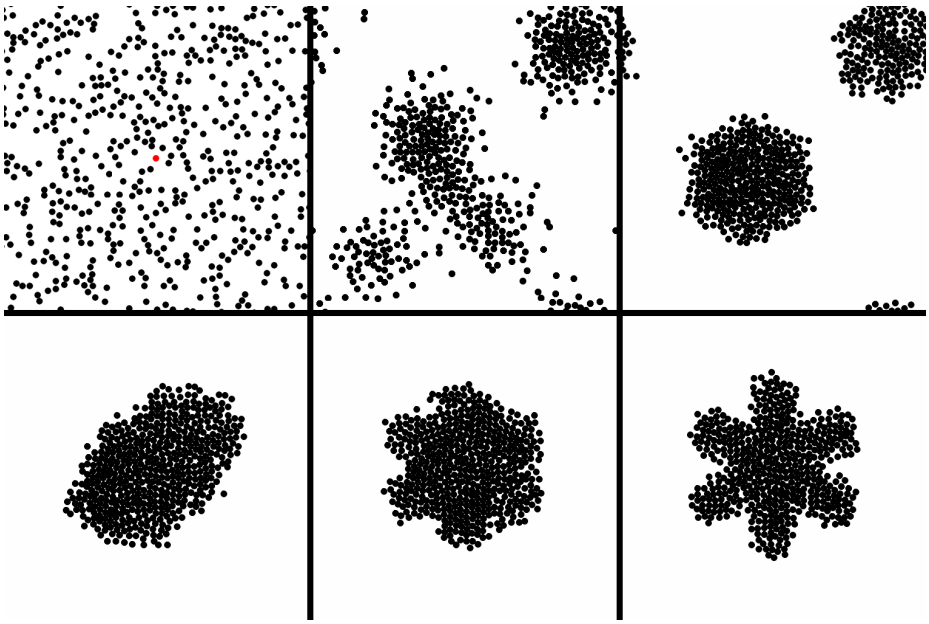


Figure 5.1: Morphogenetic Primitives self-organizing into a star shape.

configurations, we have identified for each shape, the most distinguishing macroscopic statistical moment of the configuration (final shape). It is then possible to generate random distributions of MPs that have specific statistical moments. Our work empirically shows that for bifurcating shapes one final outcome can be generated with very high probability by enforcing a constraint on the value of a single moment when generating the swarm's initial conditions. Given this feature of our system, we are able to control the final outcome of the simulation by simply thresholding the value of a statistical moment for a particular starting distribution, i.e., we constrain the initial conditions to have specified statistical properties.

5.2 Related Work

Building upon the references cited in the previous chapter for analyzing self-organizing systems, a number of efforts have attempted to control these types of systems. Pfeifer et al. point out that biology and self-organization can assist in the design of autonomous

robots [77]. Our current work takes a bio-inspired multi-agent approach to spatial self-organization and focuses on influencing or directing the outcome of such systems.

Schmickl et al. define collective perception in a robot swarm as a single map obtained by joining multiple individual robot perceptions [83]. They present two approaches to collective perception that only require individual robots with limited abilities of local perceptions. Liu et al. present an adaptation mechanism to adjust the ratio of foragers to resters in a swarm of foraging robots with the goal of maximizing the swarm's net energy income [67]. Shen et al. propose a Digital Hormone Model for controlling robot swarms that are based on local communication [88]. This model considers the topological structure of the organization and does not require a unique identifier for each individual.

In our work, based on insights about the moments of the coordinates of moving agents in a $2D$ arena, we generate biased initial random conditions that control the final outcome of a self-organizing shape formation process [4].

5.3 Biased Initial Conditions

In order to control the final outcome of a self-organizing shape formation simulation, we first extract features that capture the spatial distribution of the MPs. Moments provide a quantitative way to describe a distribution. Since MPs are defined as small discs, we use the center of each disc to represent each MP's location. We therefore can simplify the collection of MP locations as a set of $2D$ points, and apply moment analysis to this set over the duration of the MP simulation.

5.3.1 Statistical Moments

We calculate the mean (first moment), variance (second central moment), skewness (third central moment) and kurtosis (fourth central moment) from the x and y coordinates of the

MP centers. The population size of the agents is denoted as n, ($n = 500$), and the formulas for the four moments M_1 to M_4 are given in Equations 4.1 to 4.4,

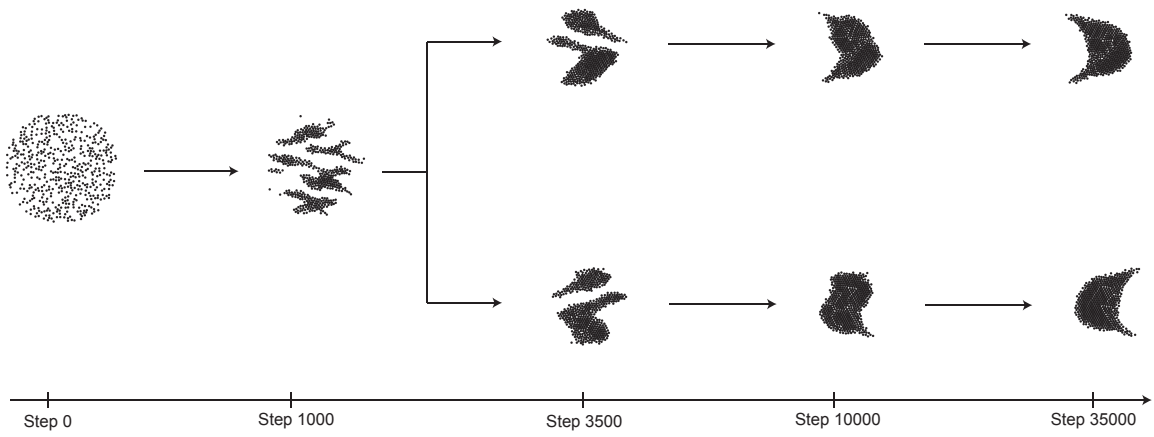


Figure 5.2: Shape aggregation of the quarter-moon MPs.

One of our aggregations, which produces what we call the quarter-moon shape, provides an example of the moment analysis. Of the 200 simulations starting with a uniform random, unbiased initial condition, 100 produce left-facing structures and 100 produce right-facing structures. Figure 5.2 shows a typical swarm aggregation for this shape. The four moments of both x and y coordinates are calculated over all $35,000$ simulation steps. Additionally the mean and standard deviation of each statistical moment is calculated for all the simulations runs in the two categories, i.e. left-facing and right-facing, over the simulation time steps. Plotting the mean and the mean $\pm$ the standard deviation of the moments over time immediately highlights the moments which are the most differentiating and may be used to identify specific shapes. For the quarter-moon example the third x moment (skewness) is the one with the greatest separation of values for the two possible outcomes, as seen in Figure 5.3. The solid lines are the mean of the skewness of the x coordinate and the dotted lines are mean $\pm$ standard deviation. The red curves are produced from structures that are right-facing and who follow the path in the top of Figure 5.2. The blue curves

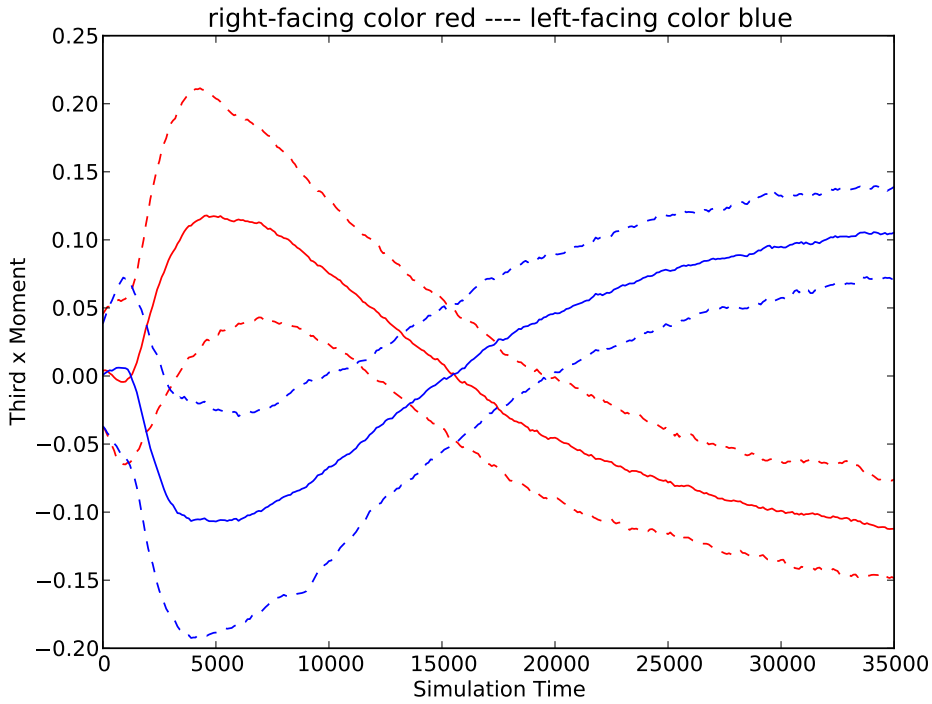


Figure 5.3: Skewness of the x coordinate of the quarter-moon shapes over time, beginning with unbiased, random initial conditions.

are produced from the left-facing structures, with a typical aggregation presented at the bottom of Figure 5.2.

By analyzing the time series in Figure 5.3, we see that the skewness of the x coordinates of the two classes starts at about the same value, approximately -0.05 to 0.05, at time step 0. The values should be near zero, since all simulations begin with uniform random configurations. The skewness of the two classes first separates by increasing or decreasing, followed by a zero crossing and then a reversed trend appears until they reach their final states at step $35,000$. Observing the values for the red and blue curves over all simulation time steps, we can identify three regions in the plot: a region occupied by blue curves only, a region occupied by red curves only and an overlapping region of the red and blue curves. To be specific, considering the values of skewness in Figure 5.3 (by projecting the curves

into the y axis), the range $[-0.192, -0.154]$ is covered by blue curves only; $[0.145, 0.211]$ is covered by red curves only and $[-0.154, 0.145]$ is covered by both colors.

5.3.2 Generating Constrained Biased Distributions

Once the most significant distinguishing moment for a shape is identified, the information is utilized to direct the shape aggregation by imposing constraints on this moment in the initial conditions. We assume that the MPs' x and y coordinates are independent, and therefore create two probability density functions, each representing the x and y coordinate. One probability density function is created for the constrained coordinate and the other coordinate is considered to be uniformly random. Samples are drawn independently from the two distributions to produce a single (x, y) location.

This approach generates random distributions, i.e. $2D$ random initial configurations for the shape formation simulation, that meet a constraint on a particular moment in the x or y coordinate. We have found that constraining one of the eight moments (mean, variance, skewness and kurtosis for x and y) is sufficient for producing satisfactory results. Constraining multiple moments does not significantly improve the outcomes. Once one significant moment of a distribution and its threshold value have been identified, the remaining three moments for that spatial coordinate (x or y) may be set to values derived from a symmetric normal distribution. These values are $mean = 500$ (the center of our computational arena), $variance = mean^2$ $(250,000$, in order to approximate a uniform random distribution), $skewness = 0$ (to make the distribution symmetric), and $kurtosis = 0$ (to make the distribution normal). Once the four moments for one of the coordinates have been specified, a probability density function (PDF) with those moments is defined. The values for the other coordinate are generated from a uniform random distribution.

For the constrained dimension we create a Gram-Charlier expansion of the normal distri-

bution [85] with specified moments. This Gaussian expanded probability density function is then discretized with 100,000 samples with values falling within the range of 0 to 1000 (the range of the computational arena). Slice sampling [74], a Markov chain sampling method, is then utilized to draw 500 samples from this discretized distribution. Theoretically, the specified PDF can be sampled to produce a distribution that has the same moments as the PDF. Our experience has shown that the sample size needs to be quite large (on the order of 1 million) for this to be true. For our sample size of 500 (the number of MPs in an aggregation simulation) and heavily biased PDFs, the resulting moments of the finite sample set do not necessarily match the ones desired for a particular shape.

Since the distribution generation process cannot guarantee that a sampled distribution will meet the required moment constraints, the four moments of the generated distributions for the constrained coordinate are computed to determine if the constraint is actually met. If the value of the significant moment is not above or below the specified threshold, the sampled distribution is rejected. If the constraint is met, the distribution is accepted as the initial conditions for the simulation computation. We have found, when generating initial conditions for a computational experiment, that in the worst case we have to generate approximately 400 sampled distributions in order to obtain 100 that meet the needed constraint. In the best case, the first 100 distributions meet the constraint. Typical values are closer to 150 generated distributions needed to obtain 100 acceptable biased initial conditions.

In general the threshold value for the constrained moment is defined as the mean value of the non-overlapping moment range for a particular shape. For example for the quarter-moon shape in Figure 5.2, we identified two non-overlapping regions of the third moment in the x dimension. See Figure 5.3. The region covered only by red curves (for right-facing

shapes) is $[0.145, 0.211]$. The mean value of this region is 0.178. In order to consistently produce right-facing quarter-moon shapes, we create $2D$ distributions with a constraint on the skewness in the x coordinate, and only accept distributions with a third moment greater than 0.178.

Since our agents are not points, but in fact are discs with a fixed radius, we maintain a distance of $2R$ (where R is the radius of a disc) between sample points, to ensure that the MPs do not overlap. Therefore, if an (x, y) pair is generated that is less than $2R$ distance to a previously generated location, it is rejected and another (x, y) pair is calculated. This process continues until a sufficient number of MP locations are generated for the initial conditions.

5.4 Results

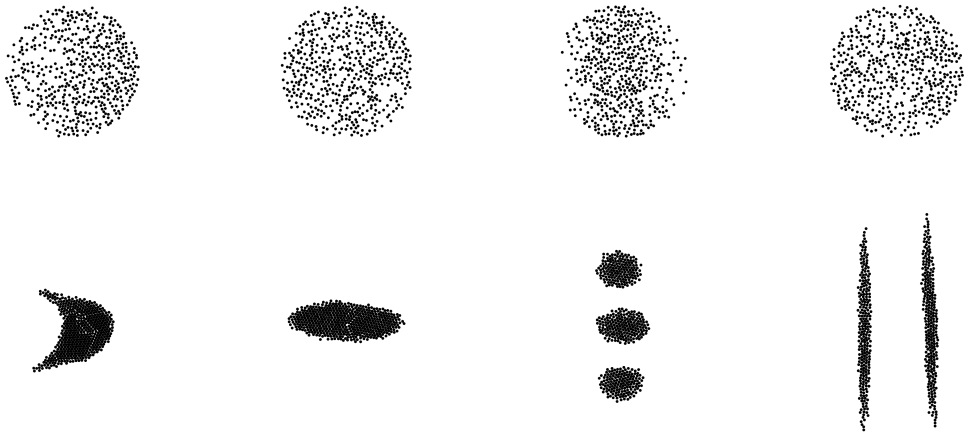


Figure 5.4: (top row) Biased initial conditions ((a) x skewness = -0.268, (b) y kurtosis = 2.150, (c) x variance = 9,596, (d) x kurtosis = 1.88) that robustly evolve (bottom row) into (a) a right-facing quarter-moon, (b) a single ellipse, (c) three discs and (d) two line segments.

We have applied the proposed method to generate initial biased configurations for a

number of bifurcating shape aggregations. These shapes include a quarter-moon, an ellipse, a set of discs, and a set of two parallel line segments. Given biased initial conditions the aggregation simulations produce one final outcome, in a large majority of, if not all, cases. Moreover, by thresholding the moment constraints on the biased initial conditions it is possible to control which shape is produced by a simulation. Figure 5.4 (top row) shows biased initial conditions for a number of shape aggregations created with this method. The bottom row illustrates the final outcome of each shape simulation that is produced from the associated biased starting configuration.

In order to identify the significant, distinguishing moments for each shape, shape simulations (usually several hundred) with unbiased initial conditions are first performed. The shapes of the final outcomes are visually inspected and placed in categories. For each final shape, the mean and standard deviation of the statistical moments of the evolving simulation runs for that shape are computed over the entire shape aggregation process. For simulations starting with uniform random, unbiased initial conditions, the final outcomes of the quarter-moon, ellipse and discs shapes evenly split into two categories, with roughly 50% of the final outcomes belonging to each class. The outcomes of the parallel line shapes are unbalanced, with 83.6% belonging to the majority class (two lines) and 16.4% belonging to the minority class (one line or 'Y' shape). These results, as well as the details that follow in the remainder of the section, are summarized in Table 5.1.

Table 5.1: Table summarizing results generated with unbiased and biased initial conditions.

Quarter-moon			
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
left-facing	50%	$Skewness_x \geq 0.178$	94%
right-facing	50%	$Skewness_x \leq -0.178$	91%
Ellipse			
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
perfect ellipse	50%	$Kurtosis_y \geq 2.11$	100%
two ellipses	50%		0%
Multiple Discs			
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
three discs	0%	$Variance_x \leq 10,270$	100%
four discs	50%		0%
five discs	50%		0%
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
three discs	0%		4%
four discs	50%	$Kurtosis_x \leq 2.09$	75%
five discs	50%		21%
Parallel Line Segments			
Shapes	Unbiased Percentage	Thresholded Moment	Biased Percentage
two lines	83.6%	$Kurtosis_x \leq 1.90$	100%
one line, "Y" shape	16.4%	$Kurtosis_x \geq 2.29$	100%

5.4.1 Quarter-moon Shape

A typical aggregation for the quarter-moon shape is shown in Figure 5.2. The simulation reaches a stable state by $35,000$ simulation steps. We identified skewness in the x coordinate to be the significant macroscopic feature for this shape. See Figure 5.3 for its evolution over the course of the aggregation. Two sets of biased initial conditions (each with 100 examples) were generated with constrained skewness values in the x coordinate, with the thresholds set as greater than 0.178 (mean of the red-only region) and less than -0.178 (a symmetric value slightly less than the mean of the blue-only region). Since the distributions are generated stochastically, they do not have the exact targeted skewness value. So our acceptance test is based on an inequality. This generates 100 biased initial conditions whose third x moment is above 0.178 and 100 biased initial conditions whose third x moment is below -0.178. We performed simulations with the quarter-moon interaction function for these 200 biased initial conditions. Of the 100 initial conditions with a thresholded third x moment below -0.178, 91% of the final outcomes are right-facing structures, 7% are left-facing and 2% produce a diagonal shape. Of the 100 initial conditions with a thresholded third x moment above 0.178, 94% are left-facing, 5% are right-facing and 1% are diagonal. Figures 5.5 and 5.6 illustrate the skewness of the x coordinates of simulations beginning with biased, random initial conditions.

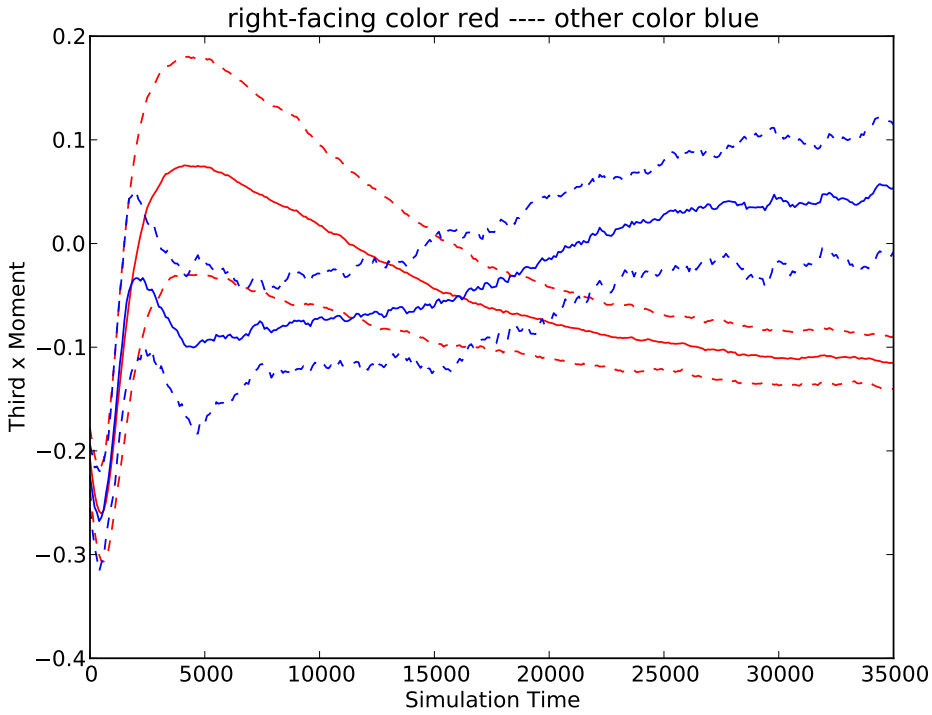


Figure 5.5: Skewness of the x coordinate of the quarter-moon shapes over time, beginning with biased, random initial conditions with skewness less than -0.178.

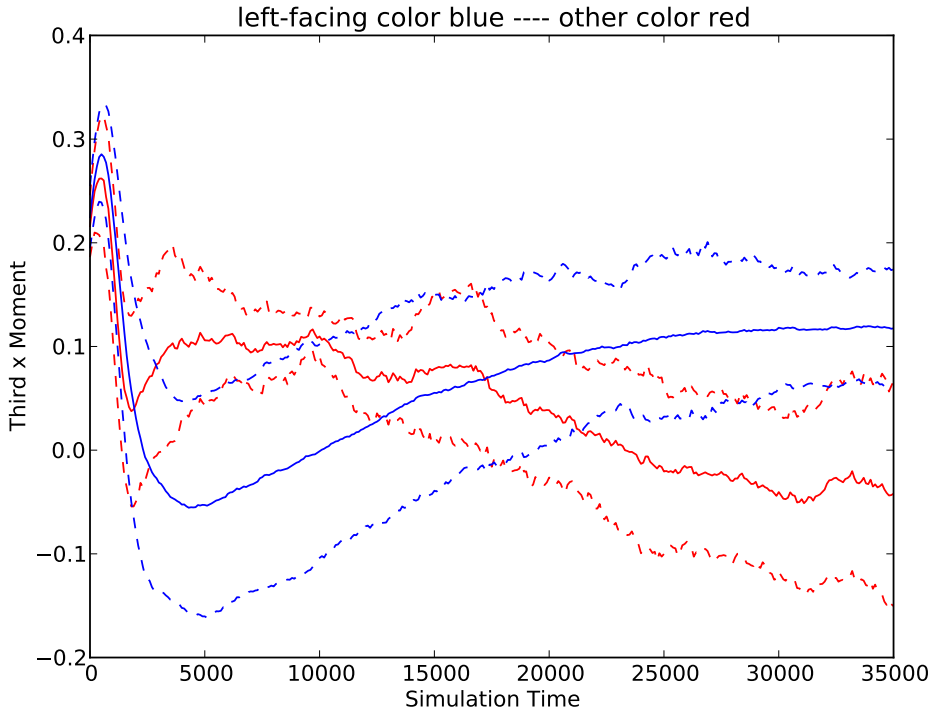


Figure 5.6: Skewness of the x coordinate of the quarter-moon shapes over time, beginning with biased, random initial conditions with skewness greater than 0.178.

5.4.2 Ellipse Shape

A typical aggregation for the ellipse shape is shown in Figure 5.7, with half of the unbiased initial conditions producing a single "perfect" ellipse, with the other half producing two ellipses or a deformed "blob". The simulation reaches a stable state by $10,000$ steps. We identified kurtosis in the y direction to be the significant macroscopic feature for this shape. See Figure 5.8. 100 simulations were performed with initial conditions that had their y coordinates' kurtosis thresholded to be above 2.11 (the mean of the blue only region). Given these biased initial conditions all simulation (100%) produced a perfect single ellipse by step $10,000$. Figure 5.9 illustrates the kurtosis of the y coordinates of simulations beginning with biased, random initial conditions.

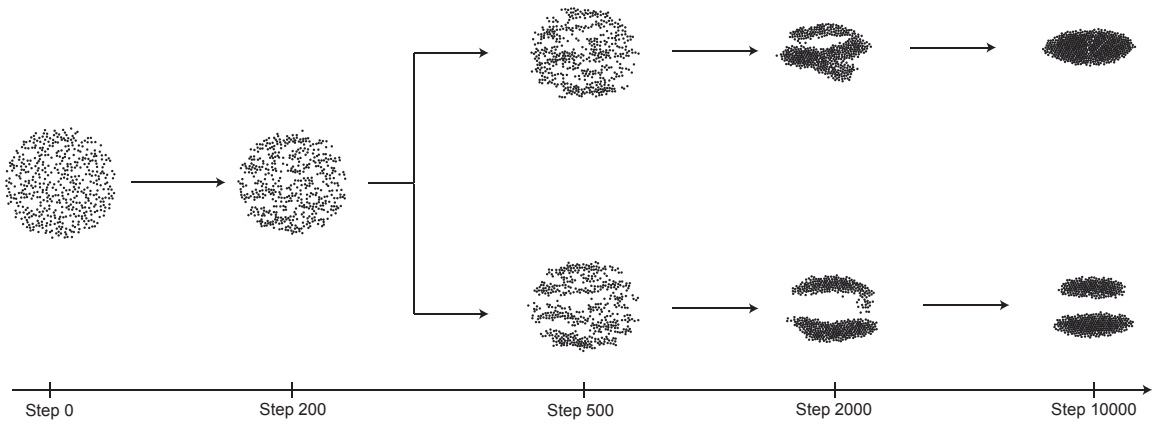


Figure 5.7: Shape aggregation of the ellipse MPs.

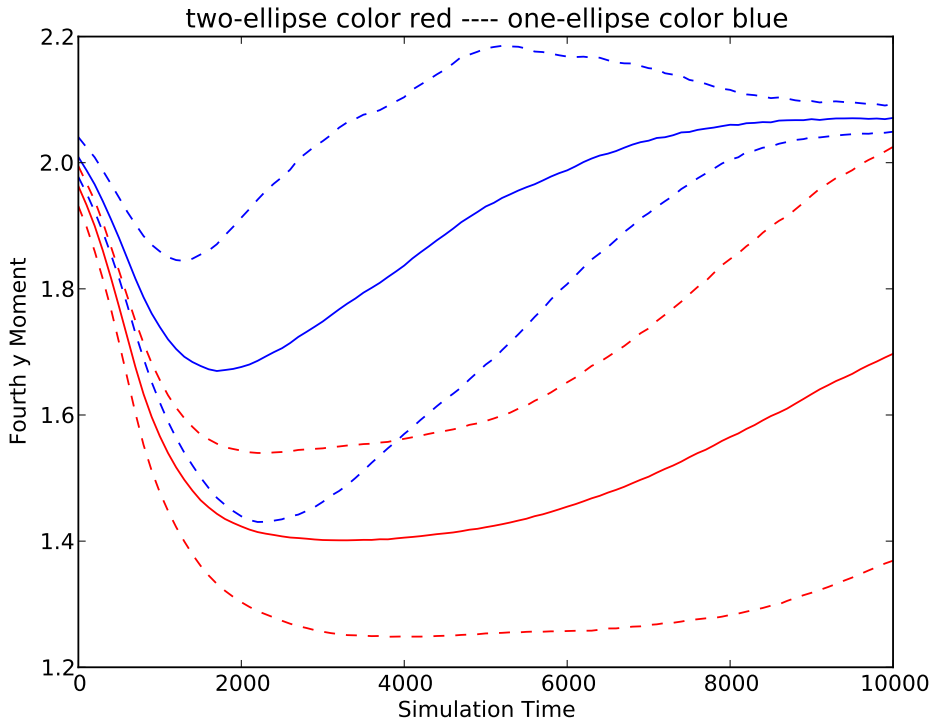


Figure 5.8: Kurtosis of the y coordinate of the ellipse shapes over time, beginning with unbiased, random initial conditions.

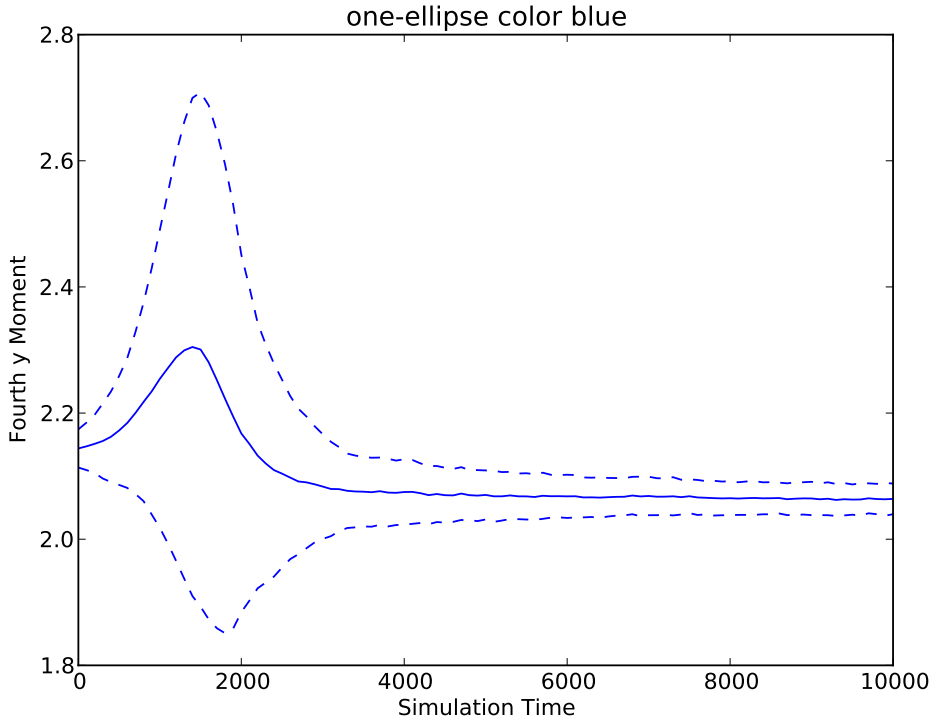


Figure 5.9: Kurtosis of the y coordinate of the ellipse shapes over time, beginning with biased, random initial conditions with kurtosis greater than 2.11.

5.4.3 Discs Shape

The discs dataset, when run with 200 unbiased initial conditions, produces 100 four discs structures and 100 structures of five or more discs, with a typical shape aggregation shown in Figure 5.10. The simulation reaches a stable state by $15,000$ steps. We identified variance in the x direction to be the significant macroscopic feature for this shape. See Figure 5.11. In our experiments as the variance of the initial conditions was lowered, we found that we could generate a new shape (one that did not appear with unbiased initial conditions), that contained only three discs. Thresholding the x variance of the initial conditions to be less than 10,270 would always (100%) produce a 3-disc result. A typical 3-disc shape is presented in Figure 5.4(c). We were unable to consistently generate a 4-disc result for most simulations by thresholding the variance. Thresholding the x kurtosis to be less than 2.09 did lead to an increased number of 4-disc results (75%), which we deemed as less than consistent or robust with 21% 5-disc structures and 4% 3-disc structures. Figures 5.12 and 5.13 illustrates the variance of the y coordinates of simulations beginning with biased, random initial conditions.

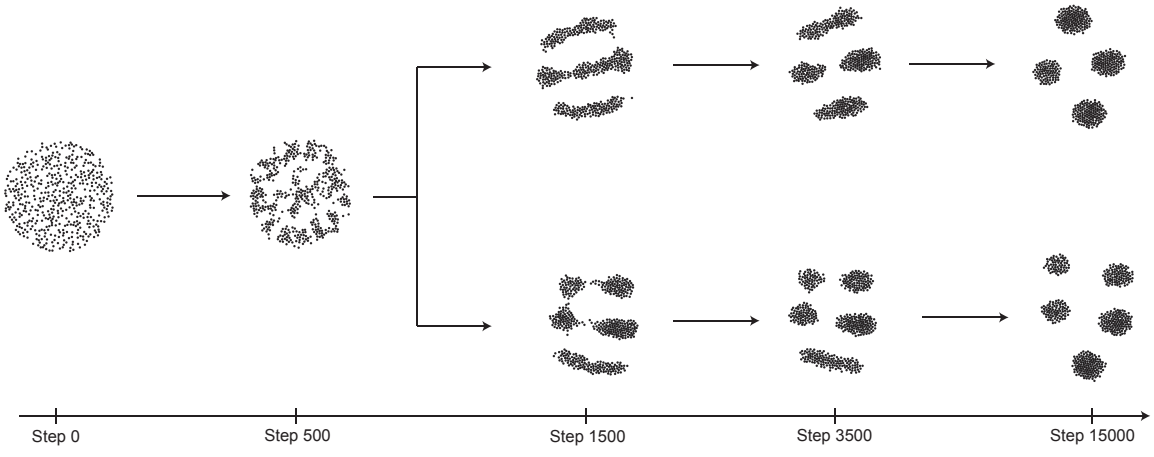


Figure 5.10: Shape aggregation of the discs MPs.

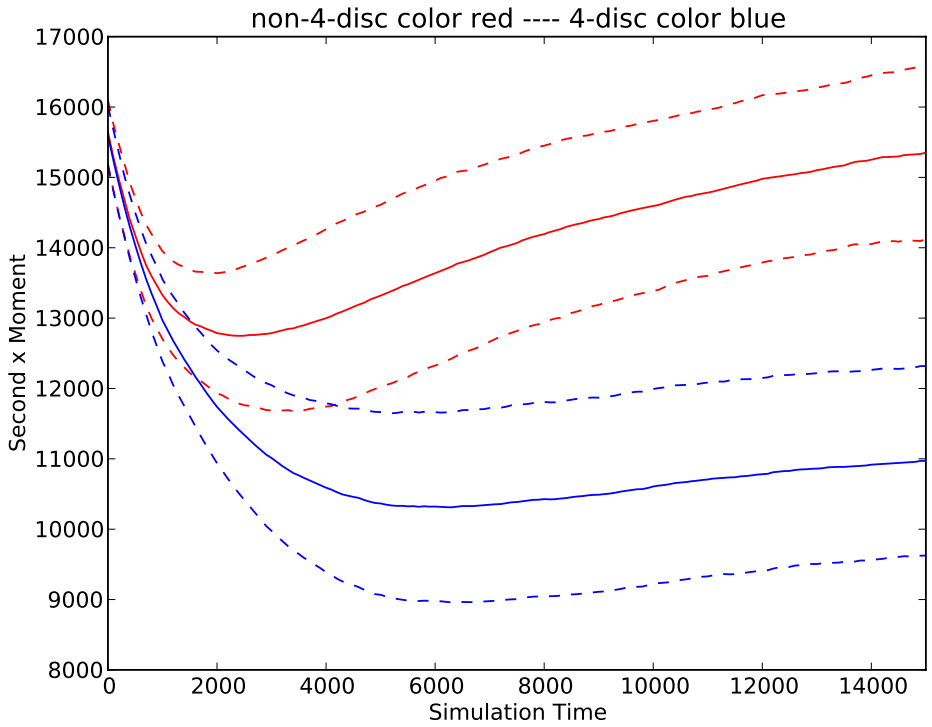


Figure 5.11: Variance of the x coordinate of the discs shapes over time, beginning with unbiased, random initial conditions.

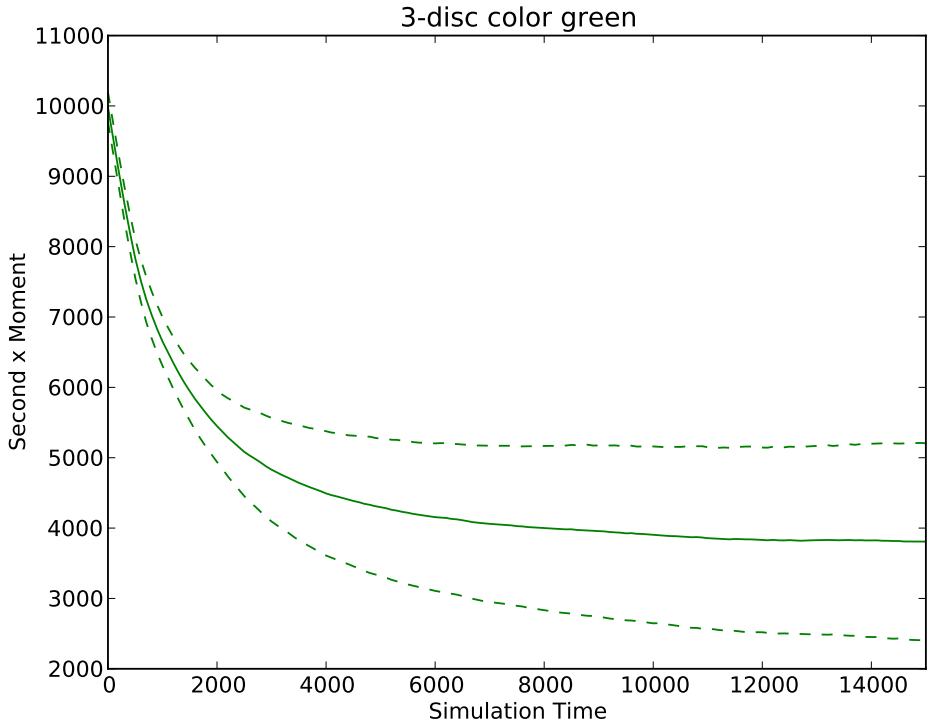


Figure 5.12: Variance of the x coordinate of the discs shapes over time, beginning with biased, random initial conditions with variance less than $10,270$.

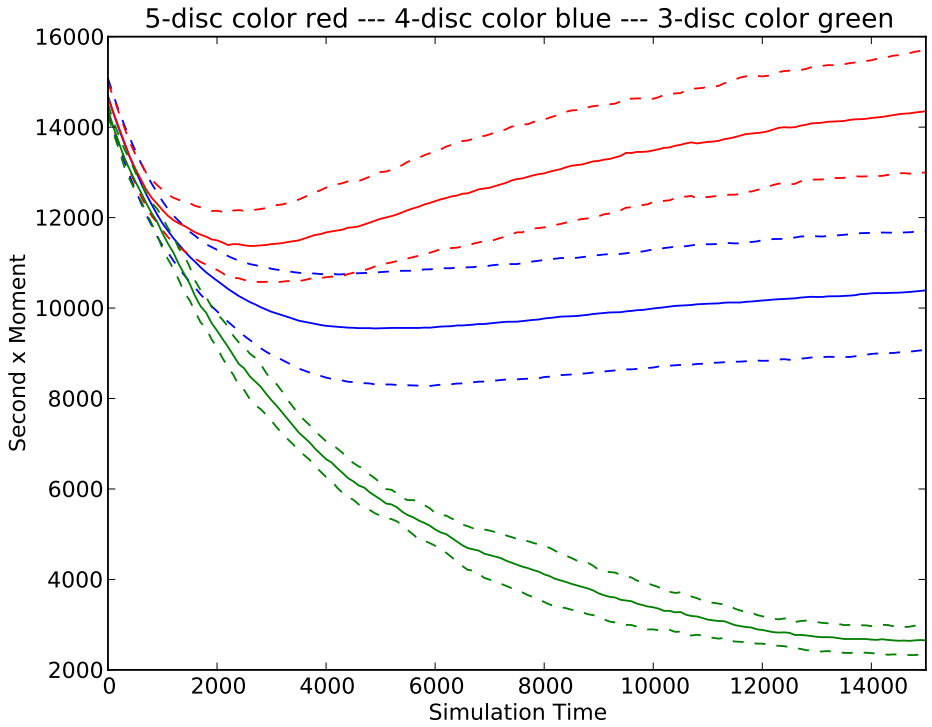


Figure 5.13: Variance of the x coordinate of the discs shapes over time, beginning with biased, random initial conditions with kurtosis less than 2.09.

5.4.4 Parallel Line Shape

The parallel line dataset, when run with unbiased initial conditions, contains 508 instances of two vertical parallel line segments (83.6%) and 100 instances of either a "Y-shape" structure or one vertical line (16.4%), as seen in Figure 5.14. The simulation reaches a stable state by $10,000$ steps. We identified kurtosis in the x coordinate to be the significant macroscopic feature for the shape. See Figure 5.15. 100 simulations were performed with initial conditions that had their x coordinates' kurtosis thresholded to be below 1.90. Given these biased initial conditions all simulation (100%) produced the two line structure. 100 simulations were then performed with initial conditions that had their x coordinates' kurtosis thresholded to be above 2.29. These biased conditions produced results that consistently (100%) created the minority class structure of a single line. For this example the x kurtosis

threshold values were found experimentally. Figures 5.16 and 5.17 illustrates the kurtosis of the x coordinates of simulations beginning with biased, random initial conditions.

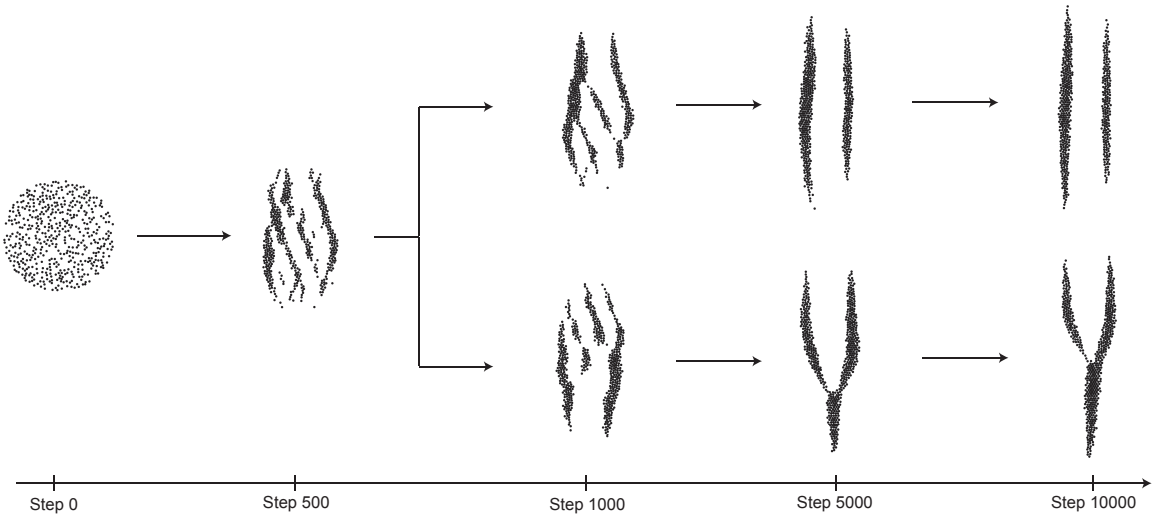


Figure 5.14: Shape aggregation of the line segment MPs.

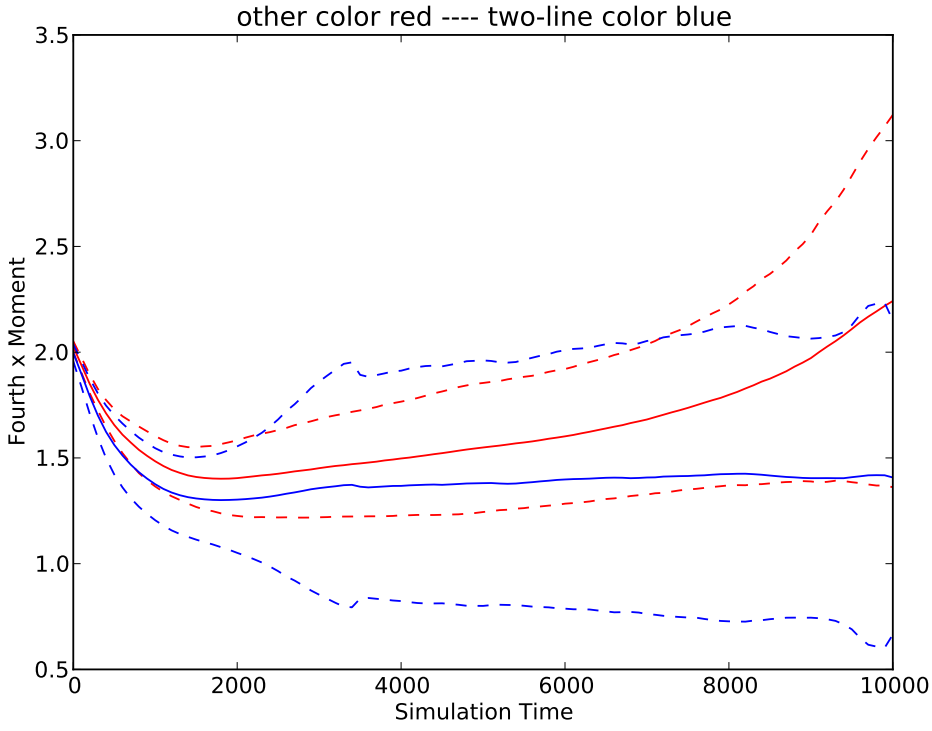


Figure 5.15: Kurtosis of the x coordinate of the line segment shapes over time, beginning with unbiased, random initial conditions.

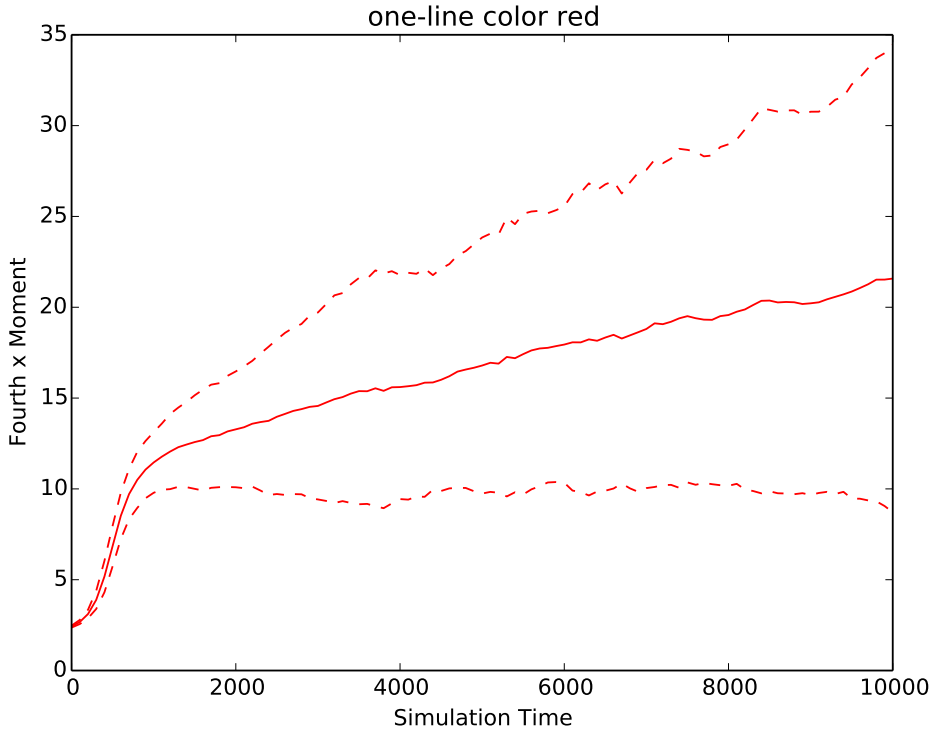


Figure 5.16: Kurtosis of the x coordinate of the line segment shapes over time, beginning with biased, random initial conditions with kurtosis greater than 2.29.

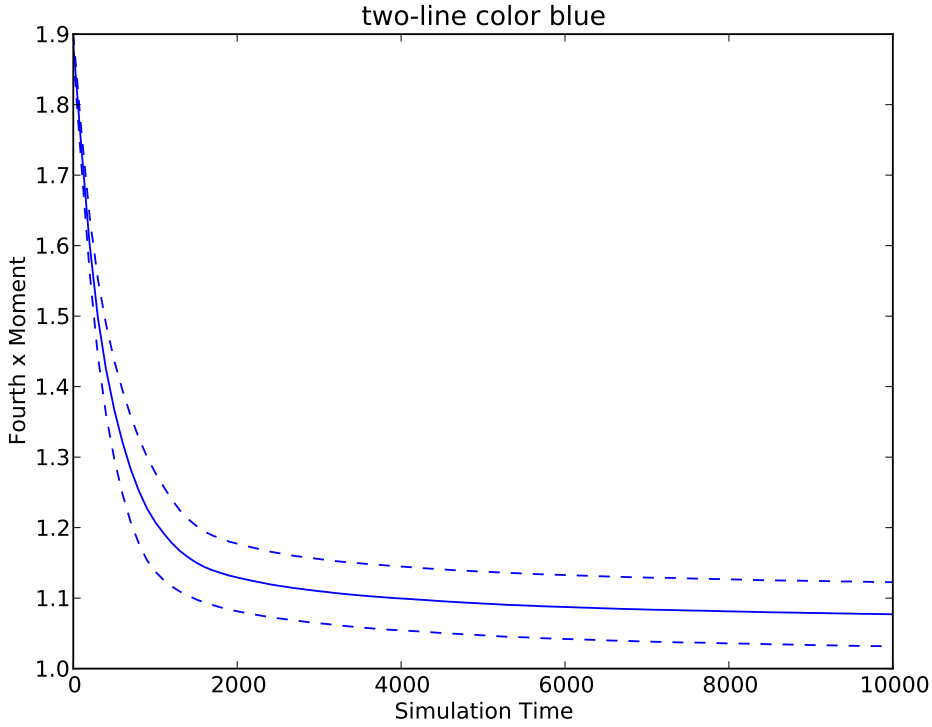


Figure 5.17: Kurtosis of the x coordinate of the line segment shapes over time, beginning with biased, random initial conditions with variance less than 1.90.

5.5 Conclusion

We have previously developed an agent-based self-organizing shape formation system. The agents perform identical behaviors based on sensing local information. Genetic programming may be used to discover local interaction rules that lead the agents to self-organize into a number of shapes. However, since the agents are initially randomly placed in the environment and they stochastically follow prescribed rules, the aggregation simulations do not always produce the same final results. We have investigated the influence of biased initial conditions on controlling the final outcomes of our system.

By analyzing the statistical moments of the agents' positions over the entire shape aggregation, we have identified significant, distinguishing moment features, and utilize them as constraints on simulation initial conditions for a number of bifurcating shapes. Biased initial conditions may be generated that meet these moment constraints, and the shape aggregations produced from them result in different shape outcomes. In some of our examples we can completely control the result of the self-organization process. In other cases we can increase the likelihood of producing a desired configuration. In general, our work shows that bifurcating shape aggregations may be directed toward producing specific shapes by simply altering their initial conditions. It demonstrates that macroscopic features may be derived from a low-level self-organization process and used to control the overall process.

As in Chapter 4, we calculate statistical moments at a global level. Furthermore, we require a global evaluation of the generated biased initial conditions. Future work will focus on eliminating global calculation and evaluation by developing a consensus coordinate system and by perturbing the agents' positions at a local level.

Chapter 6: Conclusion

We have described two approaches to spatial self-organization. Both approaches consist of primitives that utilize only local information. The interactions between the primitives are finite and are based on a chemotaxis paradigm. One is a self-sorting system consisting of a mixture of two types of agents. This system is an extension of a general cell aggregation simulation system. Given parameter values in a certain range, the system is able to consistently produce a sorted structure with one kind of cell in the center and another kind surrounding the core. The system is robust in producing the desired result, starting from many different random configurations. However, the approach is only capable of producing one type of spatial structure.

In the second effort we developed a more general shape formation system based on locally interacting shape primitives. The local interactions are defined by finite potential field functions that are evolved by an evolutionary algorithm. Given a user specified target shape, the genetic programming process automatically evolves a local field function that directs a swarm of agents to produce the shape. However, this process is not robust. Since all simulations start with initial random configurations, even given the same local potential field function, not all aggregation simulations can generate the desired final structure.

By analyzing the statistical moments of the agents' positions over the entire aggregation process, we have identified significant, distinguishing shape features. With these features, we are able to predict the final outcome of a spatial self-organization process at 5% or 10% percent of the simulation time with an overall accuracy of 81% to 91%. Furthermore, we utilize them as constraints on simulation' initial conditions for a number of bifurcating

shapes. Biased initial conditions may be generated that meet these moment constraints, and the shape aggregations produced from them result in different shape outcomes. In general, our work shows that self-organizing shape aggregations may be directed toward producing specific shapes by simply altering their initial conditions. It demonstrates that macroscopic features may be derived from a low-level self-organization process and used to control the overall process.

The contributions of the thesis include the following:

- Self-sorting of heterotypic agents. We developed a self-organizing algorithm that guarantees agent sorting. Modeling and studying cell sorting is important to developmental biology and cancer research. It also provides paradigms for biology-based algorithms of self-organization behavior.
- Morphogenetic primitives. Rather than utilizing traditional shape modeling methods (e.g. parametric shape modeling, spring-mass deformable models, solid shape modeling), we developed an 'organic' method for modeling predefined macroscopic shapes with self-organizing primitives. This approach can be extended for and applied to the control of robotic swarms, motion specification of animated crowds and generative model creation.
- Investigation of statistical moments. We studied properties of the positions of all primitives as a whole. From this we discovered macroscopic, distinguishing features that may be used to characterize different final swarm configurations. Furthermore, from these features we are be able to confidently predict the final outcome of the system at an early stage during the shape formation process.
- Directing self-organizing shape formation with biased initial conditions. The results

of the previous work lead to a control methodology that can be applied to self-organization of multi-agent shape formation, swarm robotics and sensor networks.

- Overall, this thesis describes a methodology for programming self-organizing primitives that robustly form into user-defined macroscopic shapes.

This thesis describes the first stage of a long-term effort to develop developmental-biology-inspired algorithms for spatial self-organization. Chemotaxis is the first cellular behavior we investigated as a self-organization paradigm. At this stage many open questions remain about the specifics of chemotaxis-based shape composition. Could engineering self-organizing systems benefit from other cellular behaviors? If so, what are those behaviors? More specifically, in our shape formation system, what class of shapes can be produced with the chemotaxis paradigm? Why don't the field functions always produce the same shape from initial uniformly random configurations? Are there other ways to control the final outcome of the shape formation process besides starting the simulations with biased initial conditions? What is the relationship between the form of a chemical field function and its associated final aggregated shape? Finding the answers to these questions is the core of future research.

The results presented here provide the proof of concept that paradigms for general spatial self-organization algorithms may be found by studying cell biology, specifically the intercellular mechanisms of morphogenesis. We also demonstrate that macroscopic features can be used to predict and control the outcome of spatial self-organization.

Bibliography

[1] H. Abelson, R. Weiss, D. Allen, D. Coore, C. Hanson, G. Homsy, T.F. Knight Jr, R. Nagpal, E. Rauch, and G.J. Sussman. Amorphous computing. *Communications of the ACM*, 43(5):74–82, 2000.

[2] B. Alberts, D. Bray, K. Hopkin, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Essential Cell Biology*. Garland Publishing, 2nd edition, 2003.

[3] L. Bai and D. Breen. Calculating Center of Mass in an Unbounded 2D Environment. *Journal of Graphics Tools*, 13(4):53–60, 2008.

[4] L. Bai and D.E. Breen. Directing spatial self-organization via biased, random initial conditions. In *Proc. IEEE Swarm Intelligence Symposium*, under review.

[5] L. Bai, M. Eyiyurekli, and D.E. Breen. Automated shape composition based on cell biology and distributed genetic programming. In *Proc. Genetic and Evolutionary Computation Conference*, pages 1179–1186, 2008.

[6] L. Bai, M. Eyiyurekli, and D.E. Breen. An emergent system for self-aligning and self-organizing shape primitives. In *Proc. 2nd International Conference on Self-Adaptive and Self-Organizing Systems*, pages 445–454, 2008.

[7] L. Bai, M. Eyiyurekli, and D.E. Breen. Self-organizing primitives for automated shape composition. In *Proc. IEEE International Conference on Shape Modeling & Applications*, pages 147–154, 2008.

[8] L. Bai, M. Eyiyurekli, P.I. Lelkes, and D.E. Breen. Self-organized sorting of heterotypic agents via a chemotaxis paradigm. *Science of Computer Programming*, 78(5):594–611, 2013.

[9] L. Bai, R. Gilmore, and D.E. Breen. Predicting spatial self-organization with statistical moments. In *Proc. Spatial Computing Workshop of the AAMAS Conference*, Article 2, 2014.

[10] Linge Bai and David E Breen. Chemotaxis-inspired cellular primitives for self-organizing shape formation. In R. Doursat, H. Sayama, and O. Michel, editors, *Morphogenetic Engineering*, pages 209–237. Springer, Berlin, 2012.

[11] T. Balch and R.C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[12] J. Beal. Functional blueprints: An approach to modularity in grown systems. *Swarm Intelligence*, 5(3-4):257–281, 2011.

[13] Jacob Beal, Stefan Dulman, Kyle Usbeck, Mirko Viroli, and Nikolaus Correll. Organizing the aggregate: Languages for spatial computing. *arXiv preprint arXiv:1202.5509*, 2012.

[14] J.M. Belmonte, G.L. Thomas, L.G. Brunnet, R.M.C. de Almeida, and H. Chate. Self-propelled particle model for cell-sorting phenomena. *Physics Review Letters*, 100:248702, 2008.

[15] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[16] E. Bonabeau, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz. Three-dimensional architectures grown by simple stigmergic agents. *Biosystems*, 56(1):13–32, 2000.

[17] D. Bray. *Cell Movements*. Garland Publishing, New York, 2000.

[18] C. Brown. *UNIX Distributed Programming*. Prentice Hall, New York, 1994.

[19] Samuel R Buss and Jay P Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics (TOG)*, 20(2):95–126, 2001.

[20] S. Camazine. *Self-Organization in Biological Systems*. Princeton University Press, 2001.

[21] Luca Cardelli. On process rate semantics. *Theoretical Computer Science*, 391(3):190–215, 2008.

[22] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[23] J. Crank. *The Mathematics of Diffusion*. Oxford University Press, Oxford, 2nd edition, 1975.

[24] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[25] A. Czirók, H.E. Stanley, and T. Vicsek. Spontaneously ordered motion of self-propelled particles. *Journal of Physics A: Mathematical and General*, 30(5):1375–1385, 1997.

[26] J.P. Desai, J.P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[27] R. Doursat. The growing canvas of biological development: Multiscale pattern generation on an expanding lattice of gene regulatory networks. *InterJournal: Complex Systems*, 1809, 2006.

[28] R. Doursat. Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering. In R.P. Würtz, editor, *Organic Computing*, chapter 8, pages 167–200. Springer Verlag, 2008.

[29] R. Doursat. Organically grown architectures: Creating decentralized, autonomous systems by embryomorphic engineering. *Organic Computing*, pages 167–199, 2008.

[30] R. Doursat. Programmable architectures that are complex and self-organized: From morphogenesis to engineering. *Artificial Life*, 11:181–188, 2008.

[31] R. Doursat. The self-made puzzle: Integrating self-assembly and pattern formation under non-random genetic regulation. *InterJournal: Complex Systems*, 2292, 2008.

[32] D.G. Drubin, editor. *Cell Polarity*. Oxford University Press, 2000.

[33] P. Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Proc. 4th European Conference on Artificial Life*, pages 205–213, 1997.

[34] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[35] Aguston E Eiben and Marc Schoenauer. Evolutionary computing. *Information Processing Letters*, 82(1):1–6, 2002.

[36] E. Eisenbach et al. *Chemotaxis*. Imperial College Press, London, 2004.

[37] M. Eyiyurekli, L. Bai, P. Lelkes, and D. Breen. Chemotaxis-based sorting of self-organizing heterotypic agents. In *Proc. ACM Symposium on Applied Computing*, pages 1315–1322, 2010.

[38] M. Eyiyurekli, P. Lelkes, and D. Breen. A computational system for investigating chemotaxis-based cell aggregation. In *Proc. European Conference on Artificial Life*, pages 1034–1049, 2007.

[39] M. Eyiyurekli, P. Lelkes, and D. Breen. Simulation of chemotaxis-based sorting of heterotypic cell populations. In *Proc. IEEE / NIH BISTI Life Science Systems & Applications Workshop*, pages 47–50, 2007.

[40] M. Eyiyurekli, P. Manley, P. Lelkes, and D. Breen. A computational model of chemotaxis-based cell aggregation. *BioSystems*, 93(3):226–239, 2008.

[41] B.E. Farrell, R.P. Daniele, and D.A. Lauffenburger. Quantitative relationships between single-cell and cell-population model parameters for chemosensory migration responses of alveolar macrophages to C5a. *Cell Motility and the Cytoskeleton*, 16:279–293, 1990.

[42] Giancarlo Ferrari-Trecate, Annalisa Buffa, and Mehdi Gati. Analysis of coordination in multi-agent systems through partial difference equations. *IEEE Trans. on Automatic Control*, 51(6):1058–1063, 2006.

[43] K.W. Fleischer. *A Multiple-Mechanism Developmental Model for Defining Self-Organizing Geometric Structures*. PhD thesis, California Institute of Technology, 1995.

[44] K.W. Fleischer. Investigations with a multicellular developmental model. In *Proc. Artificial Life V*, pages 389–408, 1996.

[45] K.W. Fleischer and A.H. Barr. A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. *Artificial Life III*, pages 389–408, 1994.

[46] K.W. Fleischer, D.H. Laidlaw, B.L. Currin, and A.H. Barr. Cellular texture generation. In *Proc. SIGGRAPH*, pages 239–248, 1995.

[47] R.A. Foty, C.M. Pfleger, G. Forgacs, and M.S. Steinberg. Surface tensions of embryonic tissues predict their mutual envelopment behavior. *Development*, 122:1611–1620, 1996.

[48] C. Gagné and M. Parizeau. Genericity in evolutionary computation software tools: Principles and case-study. *International Journal on Artificial Intelligence Tools*, 15(2):173–194, 2006.

[49] S.F. Gilbert. *Developmental Biology*. Sinauer Associates, Inc., Sunderland, MA, 8th edition, 2006.

[50] J.A. Glazier, R. Raphael, F. Graner, and Y. Sawadac. The energetics of cell sorting in three dimensions. In *Interplay of Genetic and Physical Processes in the Development of Biological Form*, pages 54–61. World Scientific Publishing Company, Singapore, 1995.

[51] F. Graner and J. A. Glazier. Simulation of biological cell sorting using a two-dimensional extended potts model. *Physical Review Letters*, 69:2013–2016, 1992.

[52] H. Haken. *Synergetics. An introduction. Nonequilibrium phase transitions and selforganization in physics, chemistry, and biology.* Berlin and New York: Springer-Verlag, 1983.

[53] Heiko Hamann. A reductionist approach to hypothesis-catching for the analysis of self-organizing decision-making systems. In *Proc. IEEE Conference on Self-Adaptive and Self-Organizing Systems*, pages 227–236, 2013.

[54] P. Hogeweg. Evolving mechanisms of morphogenesis: on the interplay between differential adhesion and cell differentiation. *Journal of Theoretical Biology*, 203:317–333, 2000.

[55] P. Hogeweg. Computing an organism: on the interface between informatic and dynamic processes. *Biosystems*, 64:97–109, 2002.

[56] P.E. Hotz. Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. In Kumar S. and P.J. Bentley, editors, *On Growth, Form and Computers*, pages 302–318. Academic Press, 2003.

[57] M.A. Hsieh and V. Kumar. Pattern generation with multiple robots. In *Proc. International Conference on Robotics and Automation (ICRA)*, pages 2442–2447, 2006.

[58] E. Jabbarzadeh and C.F. Abrams. Chemotaxis and random motility in unsteady chemoattractant fields: A computational study. *Journal of Theoretical Biology*, 235:221–232, 2005.

[59] E.F. Keller and L.A. Segel. Model for chemotaxis. *Journal of Theoretical Biology*, 30(2):225–234, 1971.

[60] Ioannis G Kevrekidis, C William Gear, and Gerhard Hummer. Equation-free: The computer-aided analysis of complex multiscale systems. *AIChE Journal*, 50(7):1346–1355, 2004.

[61] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:863–868, 1983.

[62] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[63] K Kruse and F Jülicher. Morphogenetic processes in animals and plants. *The European Physical Journal E: Soft Matter and Biological Physics*, 33(2):97–97, 2010.

[64] M. Kumar, D.P. Garg, and V. Kumar. Self-sorting in a swarm of heterogeneous agents. In *Proc. American Control Conference*, pages 117–122, 2008.

[65] Kristina Lerman, Alcherio Martinoli, and Aram Galstyan. A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm Robotics*, pages 143–152. Springer, 2005.

[66] Xia Lin, Dagobert Soergel, and Gary Marchionini. A self-organizing semantic map for information retrieval. In *Proc. International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 262–269, 1991.

[67] Wenguo Liu, Alan Winfield, Jin Sa, Jie Chen, and Lihua Dou. Strategies for energy optimisation in a swarm of foraging robots. In *Swarm Robotics*, pages 14–26. Springer, 2007.

[68] M. Mamei, M. Vasirani, and F. Zambonelli. Experiments of morphogenesis in swarms of simple mobile robots. *Applied Artificial Intelligence*, 18(9-10):903–919, 2004.

[69] N. Metropolis, A.W. Rosenbluth, M. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[70] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[71] J.C.M. Mombach, J.A. Glazier, R.C. Raphael, and M. Zajac. Quantitative comparison between differential adhesion models and cell sorting in the presence and absence of fluctuations. *Phys. Rev. Lett.*, 75:2244–2247, 1995.

[72] R. Nagpal. Programmable self-assembly using biologically-inspired multiagent control. In *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, pages 418–425, 2002.

[73] R. Nagpal, A. Kondacs, and C. Chang. Programming methodology for biologically-inspired self-assembling systems. In *Proc. AAAI Spring Symposium on Computational Synthesis: From Basic Building Blocks to High Level Functionality*, pages 173–180, 2003.

[74] Radford M Neal. Slice sampling. *Annals of Statistics*, pages 705–741, 2003.

[75] Andrew C Oates, Luis G Morelli, and Saúl Ares. Patterning embryos with oscillations: structure, function and dynamics of the vertebrate segmentation clock. *Development*, 139(4):625–639, 2012.

[76] M. Peifer. Developmental Biology: Birds of a feather flock together. *Nature*, 395:324–325, 1998.

[77] Rolf Pfeifer, Max Lungarella, and Fumiya Iida. Self-organization, embodiment, and biologically inspired robotics. *Science*, 318(5853):1088–1093, 2007.

[78] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM, 1987.

[79] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm robotics*, pages 10–20. Springer, 2005.

[80] J.M. Savinell, G.M. Lee, B.O. Palsson, and Ann Arbor. On the orders of magnitude of epigenic dynamics and monoclonal antibody production. *Bioprocess Engineering*, 4:231–234, 1989.

[81] H. Sayama. Swarm chemistry. *Artificial Life*, 15(1):105–114, 2009.

[82] H. Sayama. Robust morphogenesis of robotic swarms [application notes]. *Computational Intelligence Magazine, IEEE*, 5(3):43–49, 2010.

[83] Thomas Schmickl, Christoph Möslinger, and Karl Crailsheim. Collective perception in a robot swarm. In *Swarm Robotics*, pages 144–157. Springer, 2007.

[84] J.J. Schneider and S. Kirkpatrick. *Stochastic Optimization*. Springer Verlag, 2006.

[85] Skipper Seabold and Josef Perktold. Statsmodels: econometric and statistical modeling with python. In *Proc. 9th Python in Science Conference*, pages 57–61, 2010.

[86] G. Serini et al. Modeling the early stages of vascular network assembly. *The EMBO Journal*, 22(8):1771–1779, 2003.

[87] G. Serugendo, A. Karageorgos, O.F. Rana, and F. Zambonelli, editors. *Engineering Self-Organising Systems*. Springer, Berlin, 2004.

[88] W.M. Shen, P. Will, A. Galstyan, and C.M. Chuong. Hormone-inspired self-organization and distributed control of robotic swarms. *Autonomous Robots*, 17(1):93–105, 2004.

[89] Geoffroy Simon, Amaury Lendasse, Marie Cottrell, J-C Fort, and Michel Verleysen. Time series forecasting: Obtaining long term trends with self-organizing maps. *Pattern Recognition Letters*, 26(12):1795–1808, 2005.

[90] K. Sims. Artificial evolution for computer graphics. In *Proc. SIGGRAPH*, pages 319–328, 1991.

[91] K. Sims. Interactive evolution of equations for procedural models. *The Visual Computer*, 9(8):466–476, 1993.

[92] K. Sims. Evolving virtual creatures. In *Proc. SIGGRAPH*, pages 15–22, 1994.

[93] M.S. Steinberg. Adhesion in development: an historical overview. *Developmental Biology*, 180:377–388, 1996.

[94] M.S. Steinberg and M. Takeichi. Experimental specification of cell sorting, tissue spreading, and specific spatial patterning by quantitative differences in cadherin expression. *Proceedings of the National Academy of Science USA*, 91(1):206–209, 1994.

[95] K. Stoy and R. Nagpal. Self-reconfiguration using directed growth. In *Proc. 7th Int. Symp. on Distributed Autonomous Robotic Systems*, pages 1–10, 2004.

[96] K. Stoy and R. Nagpal. Self-repair through scale independent self-reconfiguration. In *Proc. IEEE/RSJ International Conference on Robots and Systems, (IROS*, pages 2062–2067, 2004.

[97] Jan Sudeikat and Wolfgang Renz. A systemic approach to the validation of self–organizing dynamics within MAS. In *Agent-Oriented Software Engineering IX*, pages 31–45. Springer, 2009.

[98] H. Tanner, A. Jadbabaie, and G.J. Pappas. Flocking in fixed and switching networks. *IEEE Transactions on Automatic Control*, 52(5):863–868, May 2007.

[99] G. Theraulaz and E. Bonabeau. Coordination in distributed building. *Nature*, 269:686–688, 1995.

[100] G. Theraulaz and E. Bonabeau. Modeling the collective building of complex architectures in social insects with lattice swarms. *Journal of Theoretical Biology*, 177:381–400, 1995.

[101] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, 5:97–116, 1999.

[102] T. Vicsek, A. Czirók, E. Ben-Jacob, and I. Cohen. Novel type of phase transition in a system of self-driven particles. *Physics Review Letters*, 75:1226–1229, 1995.

[103] A. Voss-Böhme and A. Deutsch. The cellular basis of cell sorting kinetics. *Journal of Theoretical Biology*, 263:419–436, 2010.

[104] G. Weiss, editor. *Multiagent Systems, 2nd Edition.* MIT Press, Cambridge, MA, 2013.

[105] J. Werfel. Biologically realistic primitives for engineered morphogenesis. In *Proc. 7th International Conference on Swarm Intelligence*, pages 131–142, 2010.

[106] J. Werfel, Y. Bar-Yam, D. Rus, and R. Nagpal. Distributed construction by mobile robots with enhanced building blocks. In *Proc. IEEE International Conference on Robotics and Automation*, pages 2787–2794, 2006.

[107] J. Werfel, D. Ingber, and R. Nagpal. Collective construction of environmentally-adaptive structures. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2345–2352, 2007.

[108] J. Werfel and R. Nagpal. Extended stigmergy in collective construction. *IEEE Intelligent Systems*, 21(2):20–28, 2006.

[109] S. Wiggins, editor. *Introduction to Applied Nonlinear Dynamical Systems and Chaos, 2nd Edition*. Springer, New York, 2003.

[110] S. Wolfram. Cellular automata. *Los Alamos Science*, 9(2–21):42, 1983.

[111] Wai-Tak Wong, Frank Y Shih, and Jung Liu. Shape-based image retrieval using support vector machines, fourier descriptors and self-organizing maps. *Information Sciences*, 177(8):1878–1891, 2007.

[112] Jeffrey M Wooldridge. Applications of generalized method of moments estimation. *The Journal of Economic Perspectives*, 15(4):87–100, 2001.

Appendix A: Calculating Center of Mass in an Unbounded 2D Environment

Abstract. We study the behavior of simple, 2-D, self-organizing primitives that interact and move in an unbounded environment to create aggregated shapes. Each primitive is represented by a disk and a unit point mass. In order to compare the aggregated shape produced by the primitives to other shapes, the centers of mass of the two shapes must be aligned. We present an algorithm for calculating the center of mass (COM) for a set of point masses that are distributed in an unbounded 2D environment. The algorithm calculates the centroid for each coordinate component separately by forming two "orthogonal" tubes, calculating a center-of-mass in 3-D for each tube and then projecting the 3-D COM back onto the tubes, in order to produce the 2-D COM of the points.

A.1 Introduction

We are studying automated shape composition based on self-organizing primitives. We have developed a method for discovering local interaction that direct the primitives to aggregate into a user-defined shape [7]. The primitives are randomly placed in an unbounded 2D environment and allowed to assemble by following local field gradients. The primitives' environment has toroidal topology, since the top edge of the computational arena is connected to the bottom edge, and the left edge is connected to the right edge. This allows the primitives to move and interact with other primitives across arena boundaries. This type of computational configuration is also described as having periodic boundary conditions. A central component of our self-organization method is a genetic programming process [62] that requires that we quantify the similarity of the aggregated shape with the user-specified shape [5]. In order to compute the similarity metric, the shapes must first be placed in over-

lapping areas of the computational environment. Therefore, we make the center of mass (COM) of the aggregate coincident with the COM of the user-defined shape in the center of the shape's 2-D Cartesian (image) space. In this paper, we present an algorithm for calculating the COM for a set of point masses in an unbounded 2D (toroidal) environment.

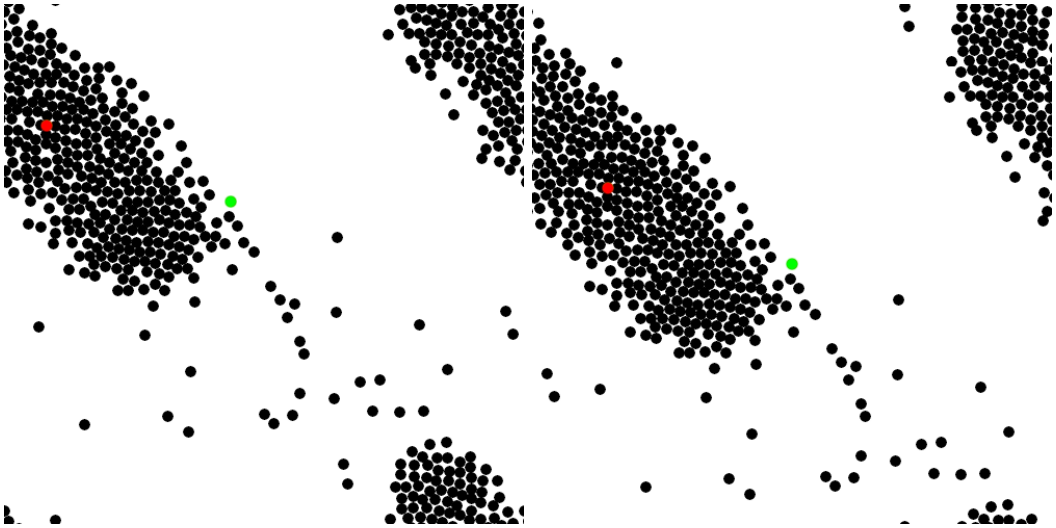


Figure A.1: (left) Original image and (right) incorrect centered result - example 1.

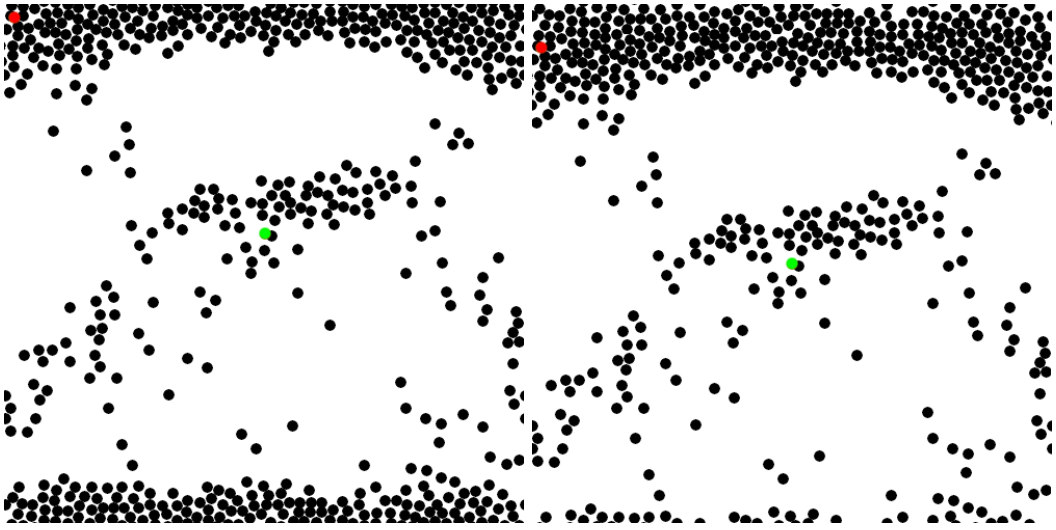


Figure A.2: (left) Original image and (right) incorrect centered result - example 2.

In general the center of mass COM of a set of point masses is calculated by the weighted

average of all the points as in the equation

$$COM = \frac{\sum m_i X_i}{\sum m_i}, \tag{A.1}$$

where m_i is the mass of point i and X_i is its location. Since our point masses exist in an unbounded environment, the problem is complicated by the lack of a proper origin for the environment's coordinate system. The specific problem we are solving involves finding the center of mass for n unit point masses in a rectangle with toroidal topology, where the origin of the finite rectangle is in the lower left corner. The COM is needed in the coordinate system of the rectangle. Equation A.1 cannot be used to provide a solution, since the collection of points may be aggregated across the environment's boundaries. For example in Figure A.1 (left), the points on the left-top edge are close to points near the top-right corner and bottom-right edge, because of the toroidal connectivity. Similarly, in Figure A.2 (left) the points along the top edge of the image are close to the points along the bottom edge of the image. Simply applying Equation A.1, using the 2-D Cartesian coordinates of the images, produces incorrect solutions, as seen in Figures A.1 (right) and A.2 (right). In these images a red dot is placed at the center of mass produced by the above method. The COM calculated by our new method is represented by a green dot, which can be found in the upper left portion of the images. When we compare aggregation results, the computed COM (red dot) is translated to the center of the image, with the pixels being rolled across edge boundaries. The desired result should place the aggregate in the center of the image. Using the 2-D Cartesian coordinates of the points along with Equation A.1 does not produce the desired outcome.

One existing solution, based on SLERP, computes a weighted spherical average [19].

The weighted average is expressed similarly to Equation A.1, but on the surface of a torus. The COM of the point masses is the point that minimizes the sum of weighted distance between COM and each point mass. However, the calculation of distance on a toroidal surface is complex and the outcome of the minimization may have multiple local solutions. Therefore, we propose a new, simpler algorithm to solve the COM problem in an unbounded 2D (toroidal) environment.

A.2 Algorithm Description

In order to properly calculate the COM of 2-D point masses distributed in an environment with periodic boundary conditions, the masses are first mapped onto 3-D tubes. A COM calculation is performed in three dimensions and then projected back into two dimensions. The algorithm calculates the centroid for each coordinate component separately by forming two tubes, and calculating a center-of-mass in 3-D for the points on each tube. The 3-D COM is then projected back onto the tubes. The location of the projected COM in the coordinate system of the tube surface is the 2-D COM of the points.

The location of a point mass in the 2-D rectangle is represented by the 2-D Cartesian coordinate (i, j), with values ranging from $(0, 0)$ to (i_{max}, j_{max}). The algorithm constructs two "orthogonal" tubes from the 2-D rectangle. We use the term *orthogonal* because one tube is formed by connecting the left edge of the rectangle with the right edge, and the other tube is formed by connecting the top edge with the bottom, conceptually creating two tubes rotated 90° from each other. The first tube ($\mathcal{T}_i$) is created by connecting the $i = 0$ edge of the rectangle with the $i = i_{max}$ edge. The second tube ($\mathcal{T}_j$) is created by connecting the $j = 0$ edge of the rectangle with the $j = j_{max}$ edge. The process of forming the tubes transforms the primitives' locations from 2-D to 3-D. The 3-D Cartesian coordinate $\mathbf{X}_k \equiv (x, y, z)$ denotes the location of the *k-th* point mass in three dimensions during these

interim calculations. The 2-D to 3-D transformation for points on tube $\mathcal{T}_i$ is defined by

$$x = r_i \cos(\theta_i), \quad y = j, \quad z = r_i \sin(\theta_i),$$
$$r_i = \frac{i_{max}}{2\pi}, \qquad \theta_i = \frac{i}{i_{max}} 2\pi. \tag{A.2}$$

The 2-D to 3-D transformation for points on tube $\mathcal{T}_j$ is defined by

$$x = i, \quad y = r_j \cos(\theta_j), \quad z = r_j \sin(\theta_j),$$
$$r_j = \frac{j_{max}}{2\pi}, \qquad \theta_j = \frac{j}{j_{max}} 2\pi. \tag{A.3}$$

Two different tubes are needed in order to calculate each coordinate of the COM. Tube $\mathcal{T}_i$ is used to calculate the i coordinate, because once transformed to cylindrical coordinates the i becomes the boundless θ_i coordinate. Tube $\mathcal{T}_j$ is used to calculate the j coordinate, because j is then mapped to θ_j.

Once the unit point masses have been mapped onto one of the tubes, the 3-D center of mass of these transformed points is calculated,

$$\bar{\mathbf{X}} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{X}_k. \tag{A.4}$$

$\bar{\mathbf{X}}$ is then projected back onto each tube. We calculate just one of the COM coordinate from each tube. $\bar{i}$ is calculated from $\bar{\mathbf{X}}$ for Tube $\mathcal{T}_i$.

$$\theta_i = atan2(-\bar{z}, -\bar{x}) + \pi, \qquad \bar{i} = \frac{i_{max}}{2\pi} \theta_i. \tag{A.5}$$

$\bar{j}$ is calculated from $\bar{\mathbf{X}}$ for Tube $\mathcal{T}_j$.

$$\theta_j = atan2(-\bar{z}, -\bar{y}) + \pi, \qquad \bar{j} = \frac{j_{max}}{2\pi}\theta_j. \tag{A.6}$$

After this calculation, point $(\bar{i}, \bar{j})$ is the center of mass of the original 2-D point masses in the 2-D Cartesian coordinate system of the rectangle. $atan2(y, x)$ is a Unix math function that computes the principal value of the arc tangent of y/x, using the signs of both arguments to determine the quadrant of the return value. It returns a value between $-\pi$ and π. The inputs to $atan2$ are negated and the output is incremented by π in order to map the output of $atan2$ to the ranges of θ ($[0, 2\pi]$), defined in Equations A.2 and A.3.

A degenerate case occurs when the 3-D center of mass lies on the central axis of a tube. In this case, the projection of the 3-D point onto the surface of the tube is not well-defined. In Equations A.5 and A.6, the degenerate case occurs for $atan2(0, 0)$. This (unlikely) situation can be detected and the output θ value can simply be defined as 0. The 3-D point is then projected to one of the edges of the 2-D rectangle. However, after rolling the 2-D rectangular image, the projected 2-D point will always be at the center of the rolled image.

A 1-D example of the algorithm is presented in Figure A.3. Point masses are given along a line representing a connected circular environment (Figure A.3a). The ends of the line are conceptually connected and marked by a star. In order to calculate the COM of the masses, the line is formed into a circle. The COM is calculated using the new 2-D positions and marked by a small red 'x' (Figure A.3b). The intermediate 2-D COM is projected back onto the circle and marked by a red disk (Figure A.3c), providing the 1-D COM in the coordinates of the line (Figure A.3d). The degenerate case of the 1-D COM calculation

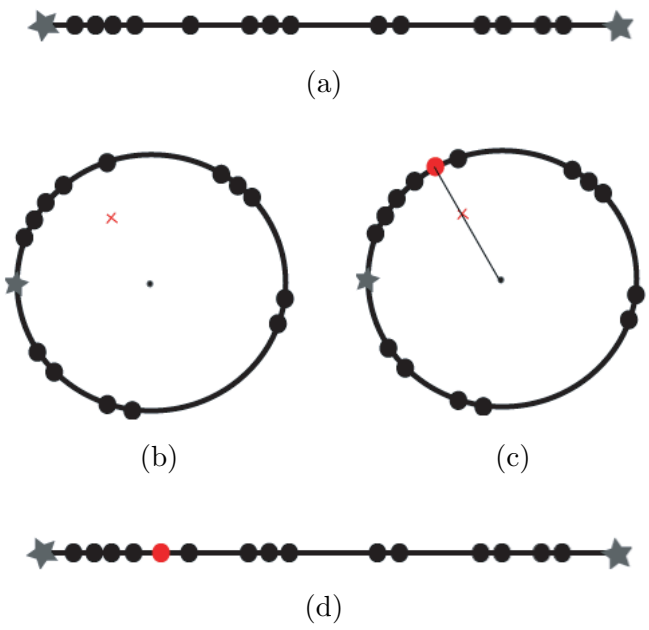


Figure A.3: Applying the COM algorithm in 1D.

occurs when the 2-D COM (red 'x' in Figure A.3b) overlaps with the center of the circle. The 2-D COM can then be projected to either end of the 1-D line.

A.3 Examples

We present four examples from our self-organization simulations that use the COM algorithm to shift aggregates, formed in an unbounded 2D environment, to the center of an image. We roll the COM $(\bar{i}, \bar{j})$ to the center of the image $(i_{max}/2, j_{max}/2)$ as seen in Figure A.4 through Figure A.7. In all of these examples, the main aggregated object crosses several boundaries of the image rectangle. In the left images the COM of the point set is displayed with a red dot. In the right images, the COM (red dot) is shifted to the center of the image rectangle, with the same shift applied to all of the points. If the shift makes an individual point cross an image edge, the point is placed on the opposite side of the images, i.e. the image is rolled.

We have found the COM algorithm to be effective and robust. It has been employed to calculate the center of mass for approximately 100,000 diverse test cases, and has always produced satisfactory results in practice.

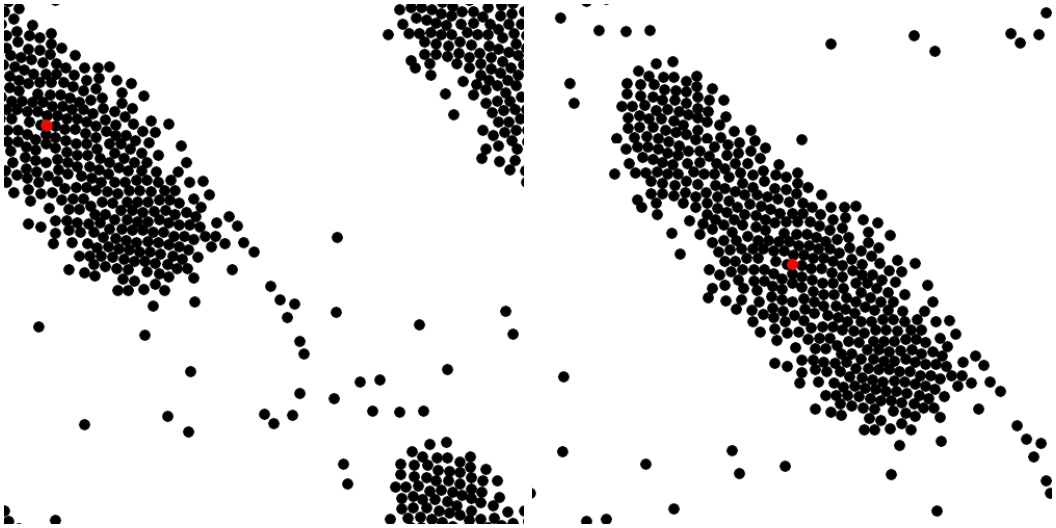


Figure A.4: (left) Original image and (right) desired centered result - example 1.

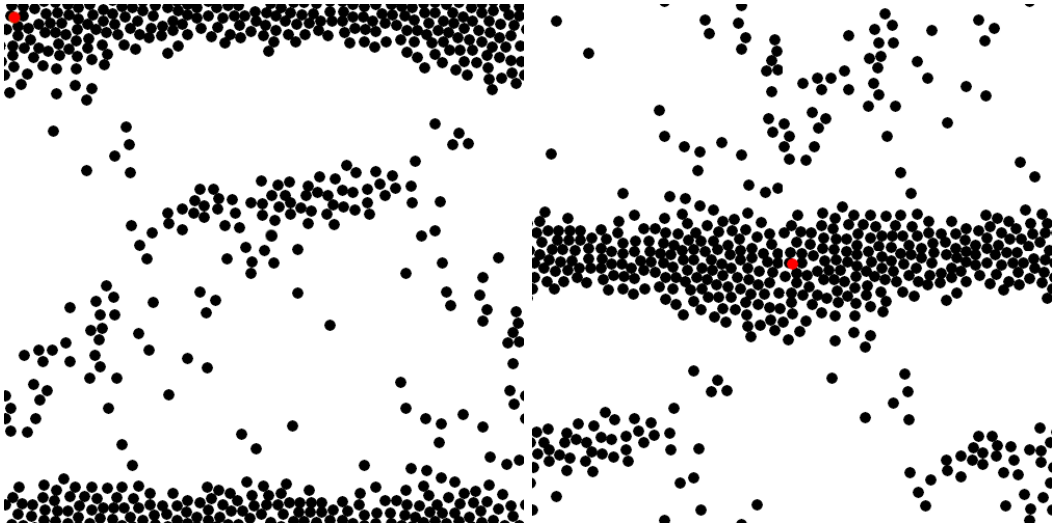


Figure A.5: (left) Original image and (right) desired centered result - example 2.

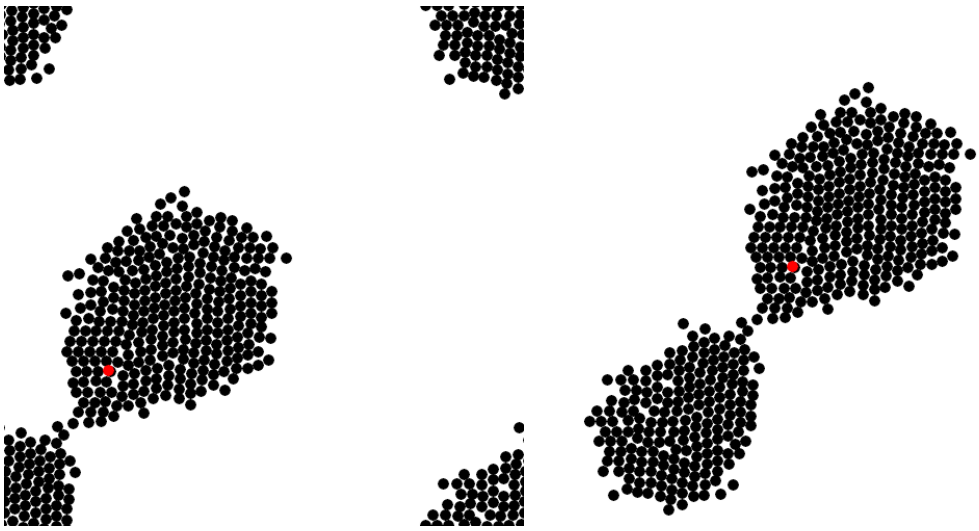


Figure A.6: (left) Original image and (right) desired centered result - example 3.

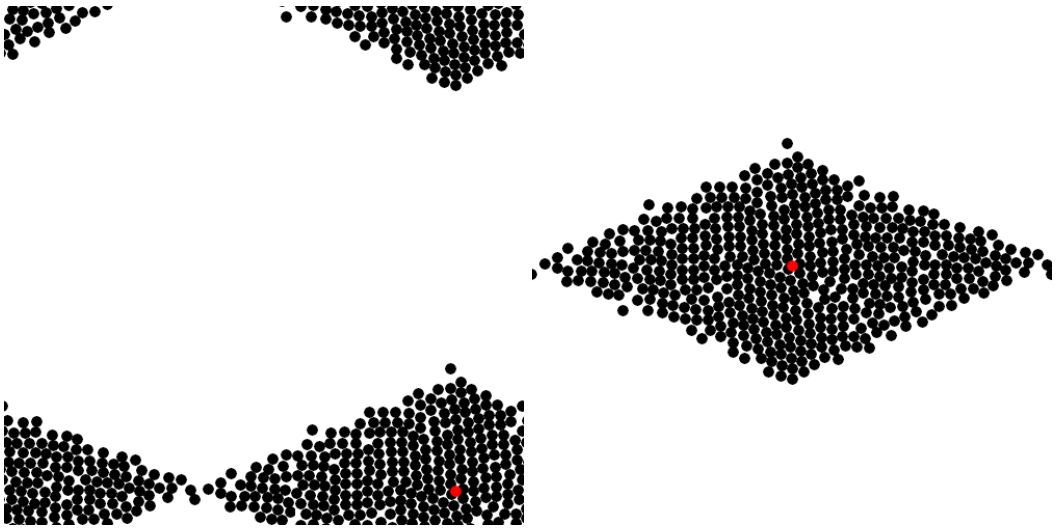


Figure A.7: (left) Original image and (right) desired centered result - example 4.

Vita

Linge Bai received her Bachelor of Engineering in Computer Science and Technology from Wuhan University in June 2006, and her Master of Science in Computer Science from Drexel University in September 2008.

Publications

- L. Bai and D.E. Breen, "Directing Spatial Self-Organization via Biased, Random Initial Conditions."

- L. Bai, R. Gilmore and D.E. Breen, "Predicting Spatial Self-Organization with Statistical Moments," *Proceedings of Spatial Computing Workshop of the AAMAS Conference*, Article 2, May 2014.

- L. Bai, M. Eyiyurekli, P.I. Lelkes and D.E. Breen, "Self-Organized Sorting of Heterotypic Agents Via a Chemotaxis Paradigm," *Science of Computer Programming*, Vol. 78, No. 5, pp. 594 – 611, May 2013.

- L. Bai, T. Widmann, F. Jülicher, C. Dahmann and D.E. Breen, "3D Surface Reconstruction and Visualization of the Drosophila Wing Imaginal Disc at Cellular Resolution," *Proceedings of SPIE-IS&T Conference on Electronic Imaging*, Volume 8654, Article 86540D, February 2013.

- L. Bai and D. Breen, "Chemotaxis-Inspired Cellular Primitives for Self-Organizing Shape Formation," R. Doursat, H. Sayama, O. Michel (eds.), *Morphogenetic Engineering: Toward Programmable Complex Systems*, Springer, Berlin, Chapter 9, 2012, pp. 209 – 237.

- D.E. Breen, T. Widmann, L. Bai, F. Jülicher and C. Dahmann, "Epithelial Cell Reconstruction and Visualization of the Developing Drosophila Wing Imaginal Disc," *Proceedings of IEEE Symposium on Biological Data Visualization*, pp. 77 – 84, Seattle, WA, October 2012.

- M. Eyiyurekli, L. Bai, P. Lelkes, D. Breen, "Chemotaxis-based Sorting of Self-Organizing Heterotypic Agents," *Proceedings of the 2010 ACM Symposium on Applied Computing*, pp. 1315 – 1322, Sierre, Switzerland, March 2010.

- L. Bai, D. Breen, "Calculating Center of Mass in an Unbounded 2D Environment," *Journal of Graphics Tools*, Vol. 13, No. 4, 2008, pp. 53 – 60.

- L. Bai, M. Eyiyurekli, D. Breen, "An Emergent System for Self-Aligning and Self-Organizing Shape Primitives," *Proceedings of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pp. 445 – 454, Venice, Italy, October 2008.

- L. Bai, "Self-Organizing Primitives for Automated 2D Shape Composition," M.S. Thesis, Drexel University, Philadelphia, PA, August 2008.

- L. Bai, M. Eyiyurekli, D. Breen, "Automated Shape Composition Based on Cell Biology and Distributed Genetic Programming," *Proceedings of Genetics and Evolutionary Computation Conference (GECCO)*, pp. 1179 – 1186, Atlanta, GA, July 2008.

- L. Bai, M. Eyiyurekli, D. Breen, "Self-Organizing Primitives for Automated Shape Composition," *Proceedings of IEEE International Conference on Shape Modeling and Applications (SMI)*, Stony Brook, NY, pp. 147 – 154, June 2008.