Amorphous Shape Mapping

A Thesis
in TCC 402

Presented to

The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

by

Christopher Frost

May 7, 2004

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in TCC Courses.

Signed: _____

Approved _____
Technical Advisor — David E. Evans

Approved _____
TCC Advisor — Patricia C. Click

Contents

List of Figures

Abstract

Research in amorphous computing studies asynchronous, identically programmed, and decentralized agents performing computations. Research in this area has produced methods for taking existing descriptions of arbitrary shapes and amorphously regrowing the approximated shape. These methods assume descriptions of shapes to be in a form accessible to traditional computers; however, using traditional computers to produce such descriptions of shapes in the physical world is a problematic and generally difficult task.

The objectives of this thesis project were to develop and analyze a method of generating a description, accessible to traditional computers, of an arbitrary two-dimensional shape by amorphously mapping the desired shape. Three interesting types of shape descriptions generatable from an amorphous shape mapping computer, connectivity, point sampling, and polygonal, are presented and discussed. With intended descriptions and uses in mind, the developed method's assumptions are stated and discussed and the primitive cell actions are given. The developed cell program's three stages are then detailed: placing the cells, mapping the region, and transferring the gathered data to a transferring computer. This thesis project's focus is mapping the given region.

Simulations of the three types of shape descriptions are described and results presented and described. Analytical results were derived for ensuring complete region description. The developed amorphous shape mapping method is able to accurately map the tested shapes using relative cell location information obtained as cells receive messages from other nearby cells. Experiments show that even cells without relative location sensing abilities can produce descriptions of the mapped shapes.

Chapter 1

Introduction

1.1 Purpose Statement

Comparing biological and computer systems with each other, one notices how capable biological systems are at coping with changes in the environment. Research in Amorphous Computing [1] over the past decade has taken ideas from the field of biology to help design systems of hundreds of thousands of agents which, through locally-specified agent behaviors, exhibit a predictable system-level behavior. The purpose of this project was to develop a method of producing maps of two-dimensional shapes accessible to traditional computers making use of, and extending, ideas from biology and amorphous computing. The developed method, described in Chapter 3 and analyzed in Chapter 4, is able to map shapes and transfer this information to traditional computers to describe the shape in terms of connectivity, sampling points, and a polygonal bound.

1.2 Problem Statement

Biological systems, such as an ant colony, assemble into complex structures and often exhibit global behavior without centralized control. Traditional computers, in contrast, generally operate through centralized control and are intolerant of small failures. Bringing biological systems' robust and distributed properties to computing has been the aim of work over the past decade by researchers developing Amorphous Computing [1] and Swarm Programming [15]. Kondacs, Chang, Frost, and Bloom of MIT have developed an amorphous computing approach for

assembling two-dimensional shapes, using decentralized, identically-programmed agents [5, 21, 25]. Shape descriptions based on origami [24] and growing points [9] have also been explored. However, all approaches have relied upon the shape already being represented in a traditional computer, using this representation to program these agents. This thesis project developed a decentralized approach for mapping unknown, arbitrary, two-dimensional shapes to produce a description accessible to a traditional computer.

1.2.1 Project Context

In discussing amorphous computing research it is helpful to define a few terms:

- *Cell:* A model of an entity (e.g. a biological cell, ant, small robot, or sensor), vulnerable to death and with limited computing power, that can communicate with nearby neighbors.
- *Amorphous Computer:* A myriad of cooperating, identical, dynamically interconnected cells whose local cooperations result in large-scale predictable behaviors. These cells also have no global knowledge of topology or cell locations.
- *Amorphous Computing:* An approach to performing computations using a system describable as an amorphous computer [1].

Forming and maintaining shapes using an amorphous computer has received much research attention. Differentiation of shape shows that even though cells are identical in the beginning, they are able to differentiate themselves in function and achieve a global behavior. Kondacs, Chang, Frost, and Bloom have developed an amorphous approach to growing two-dimensional shapes using self-assembling cells. In their model a cell assembles other cells, which in turn assemble other cells, eventually forming a disc. From this disc other discs are grown, repeating this process using the program contained within each cell. The result is a network of discs that approximate the original shape and which are able to re-grow if cells or entire portions of the shape are destroyed. The program running on all cells is generated by compiling the shape's description, a network of discs that fills an area approximating the true entity's shape. An example of this method is shown in Figure 1.1.

Nagpal developed an amorphous approach to growing two-dimensional shapes using a language based on origami. From a description that describes the folding of surfaces, a compiler

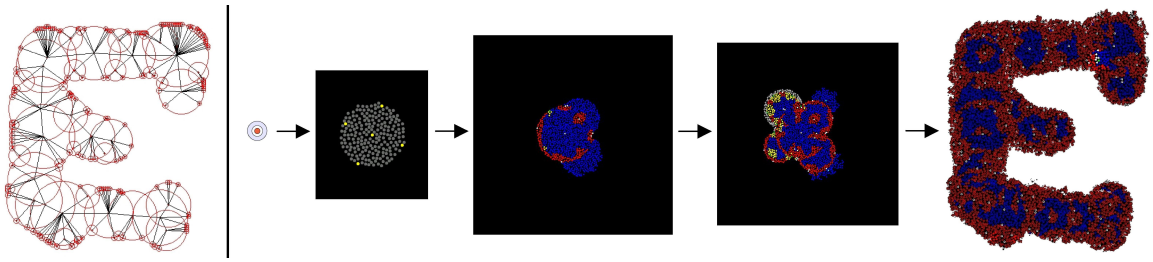

Figure 1.1: Amorphous Shape Growing Steps

Kondacs, Chang, Frost, and Bloom's shape formation using self-assembling, forming the shape of the letter *E*. Left of the divider, the network of discs describing the shape. To the right of the divider, an amorphous computer regrowing the mapped *E*, from a single cell, to a disc of cells, to a disc forming other discs, to discs forming discs, to the shape *E*.

generates an amorphous program that grows the described shape [24]. Coore developed an amorphous approach using a language based on growing points and tropisms [9].

Sensor networks and amorphous and swarm computing share many of the same design goals. While both aim for large numbers of networked agents in dynamic environments, amorphous and swarm computing differ in that their inspirations tend to be based on biological systems. There has been work in routing and event detection, but this research, such as the Cricket system by Priyantha et al at MIT [30], has generally assumed the existence of a small number of beacon nodes, which could fail and would require special manufacturing and placement. However, some research on distributed systems without special beacons has taken place. Work such as Simic and Sastry's of Berkeley on distributed location systems [33] and Wood and Stankovic's of UVa on avoiding hazardous areas [37], by creating borders of sensor nodes, may provide an interesting base for the amorphous mapping of shapes. However, both of these works give only very coarse shape descriptions as a by-product of other tasks.

Outside of shape formation, Coore [7] and later Nagpal and others [26, 27] have developed amorphous approaches to building coordinate systems. Newton and Beal have developed a method for implementing automata using an amorphous computer and implementing a functional programming language on top of this [28].

1.2.2 The Problem: Amorphously Mapping Unknown Shapes

This thesis project devised a method of building a shape's description, accessible by a traditional computer, by amorphously mapping a two-dimensional shape. With this problem

solved, the description of the shape can be made accessible to a traditional computer, where it can be transformed into other languages of shape description and analyzed or used to create cells to regrow the original shape using existing amorphous shape growing methods. Additionally, the developed method's accuracy in describing shapes and its robustness to cell failures, among other measures, has been analyzed.

1.3 Uses of Amorphous Shape Mapping

Three interesting types of descriptions can be generated from an amorphous shape mapping computer:

1. Connectivity of shapes in the region
2. The shape in terms of sampled points in a plane
3. The shape as a polygon

Chapter 4 describes and analyzes the ability of the developed method to enable the generation of these three types of descriptions.

Such shape descriptions can be taken advantage of by work in computer vision, automated mapping, and many other areas. Amorphous shape mapping could be used as a safer alternative to X-Ray usage, to map areas traditionally thought to be too hazardous for even machines, to map planet surfaces, to locate lost items, or to draw in resources from another amorphous computer to a discovered resource or warn an amorphous computer of a hazard. Section 5.2 discusses these specific uses further, as well as discussing social and ethical ramifications of such uses.

1.4 Overview

This report begins with a review of previous work in amorphous computing, swarm programming, and sensor networks. Additionally, background biological information is given in the report as its illustration gives insight to the developed methods or assumptions. Chapter 3 presents the method's assumptions, a cell program's primitive actions, and builds the developed method atop these. Next, Chapter 4 presents and discusses the methods used for and results of testing the developed method. Chapter 5 concludes the report, summarizing and interpreting the

results, discussing the social and ethical context, and discussing the future directions of this project's work.

Chapter 2

Background

This section of the proposal briefly reviews past work on related problems and should aid the reader in better understanding the concepts behind this project. The literature behind this thesis falls into two categories: amorphous and swarm computing and sensor networks.

2.1 Amorphous and Swarm Computing

Much of engineering, and especially computer science, has tended to depend upon underlying foundations being perfectly functional. Amorphous computing, begun by Gerry Sussman, Harold Abelson, and Tom Knight, Jr., is a project with the aim to develop principles and abstractions for using myriads of non-centrally controlled devices without this requirement of perfection. Swarm Programming has similar goals, focusing on systems where agents move as opposed to being static [10, 11]. The aim of amorphous computing and swarm programming is to develop models of organization allowing global behaviors to come from localized behaviors and abstractions of these models to allow programs to be composed without knowledge of the underlying system, all the while making use of properties sought after through amorphous computing. Many of these models are based on biological systems, which exhibit properties very similar to those sought after in amorphous computing, especially developmental biology [34, 35] and processes such as morphogenesis [4].

The first stage in developing amorphous computing is thus creating systems capable of communicating and organizing, developing models which will serve as foundations for further development, and assessing these. Coore, Nagpal, and Weiss of MIT's Project MAC began this

research, as microfabrication and nanotechnology were on the horizon, studying group structures in amorphous computers [8]. Research on establishing coordinate systems followed, forming the idea of using gradients to propagate messages using only local interactions [7, 26]. While these ideas have already allowed for higher level development, research in coordinate systems is still a cutting edge area [27].

As this first stage began to offer a base to build abstractions upon, research making use of this base began to tackle higher levels of amorphous computers. Several approaches taking advantage of computer science, to manage complexity, as well as biological models, to achieve robustness, were developed [25], building targeted languages containing a small set of primitive operations to describe shapes and topology. These include Coore's metaphor of growing points and tropisms [9], Nagpal's metaphor of paper folding [24], and Kondacs, Chang, Frost, and Bloom's representation as networks of discs [21]. An example of building on earlier work, the networks of discs method implements programs described in its language using replication, gradients, and competition among cells.

All approaches to describing shapes to date have relied upon the shape already being represented in a traditional computer, using this representation to program the cells which will regrow the original shape. The aim of this thesis was to develop a method of *amorphously mapping* arbitrary two-dimensional shapes, building on previous research's ideas for organization and the abstractions developed for regrowing shapes.

2.2 Sensor Networks

Sensor networks and amorphous and swarm computing, begun in the last decade, share many of the same design goals. While both aim for large numbers of networked agents in dynamic environments, amorphous and swarm computing differ in that their inspirations tend to be based on biological systems. There has been work in routing and event detection, but this research, such as the Cricket system by Priyantha et al at MIT [30], has generally assumed the existence of a small number of beacon nodes, which could fail and would require special manufacturing and placement. Some research on distributed systems has taken place, however. Work such as Simic and Sastry's of Berkeley on distributed location systems [33] and Wood and Stankovic's of UVa on avoiding hazardous areas [37], by creating borders of sensor nodes, may provide an interesting

base for the amorphous mapping of shapes. However, both of these works give only very coarse shape descriptions as a by-product of other tasks.

Chapter 3

Mapping Method

This chapter begins with a description of an example method for using an amorphous computer to map shapes that will serve as motivation for more explicitly looking at the assumptions a method makes. The developed method's assumptions about the environment, cells, and observer are then stated and discussed. The primitive actions of a cell, which follow from the method's assumptions, are then given. The developed model, building on these primitives and making use of the given assumptions, is then described.

3.1 An Example Method

One approach to using an amorphous computer to map a shape would be:

1. Drop a collection of cells over the region where the shape is likely to be
2. Upon landing, each cell:
   (a) Broadcasts a message to all other cells, stating it sent this message and whether it is on or not on the shape
   (b) Listens for messages sent by others, also recording the origin of each message, until it has heard from all cells in the collection
3. An observing computer can then retrieve any one cell and query it for the relative positions of all the other cells, producing a description of the shape

This approach would produce a description of the shape, but only when the approach's implicit assumptions are met. Let us examine some of these assumptions and see what implications they have for implementations.

The first action a cell must be able to do is sense whether it is on the shape or not. With this information it can then broadcast to other cells its status. Sensing whether a cell is on the shape seems feasible, for example, readily available small and simple light intensity sensors would detect whether or not they were on top of a computer monitor. Moving on, however, two assumptions especially stand out as severely limiting the usefulness of this approach: broadcasting data to *all cells* and requiring all cells to hear broadcasts from all other cells to know the sensing is complete.

Broadcasting a message to all other cells requires a cell to be able to send a strong enough a signal to cover the entire region being explored. Thus, cells would either need to be customizable for differently sized regions, or if they are not customizable, the size of the explorable region would be limited to a fixed size. Additionally, the number of messages a cell must process and store scales linearly with the number of cells used, increasing the amount of energy required and either processing speed or time spent sensing.

Requiring a cell to hear from all other cells implies all other cells *will* send a message, that no cells will fail. As probability shows, the chance of a single failure, P, of k independent systems is exponentially related to k and the probability a given system will fail, p: $P = p^k$ [32].

3.2 Assumptions

Section 3.1's example amorphous shape mapping computer illustrated the importance a method's assumptions are to the method's usefulness and serves as motivation for explicitly stating and assessing the assumptions of this thesis' developed method. These assumptions follow from the desired applications of an amorphous mapping computer, using large numbers of inexpensive cells to gather information about the desired shape, later describing the explored region to a traditional computer.

3.2.1 Environment

Assumption 1 *During the time cells are sensing the environment, the environment remains static. More precisely, within the time cells are sensing, it does not matter when a cell determines:*

1. *Whether or not it is on the shape*

2. *Its relative location to another cell*

This assumption removes the need to cope with data recorded at different times. For example, cells which move and come into contact with different cells throughout the sensing period, or, for cells which might float away from the shape. Further, many shapes are static, such as a plate, or may change slowly compared to the rate of mapping.

Assumption 2 *Cells can be transported to the region containing the shape to be mapped.*

Assumption 3 *The shape to be mapped is two-dimensional.*

No other assumptions about the shape, for example a need to be closed, connected, or convex, are needed for the developed cells. The method used to reconstruct the mapped shape from the data queried from the amorphous computer may, however, impose additional environmental assumptions. For example, the polygonal shape description used in Section 4.4's analysis assumes there exist no holes in shapes.

3.2.2 Cell

Assumption 4 *A cell can sense whether or not it is on the shape to be mapped with perfect reliability.*

An example of a small and inexpensive sensor, for light, was given in Section 3.1. The requirement for perfect sensor reliability may be possible to remove depending upon what information the observer wishes to rebuild and whether small errors are acceptable. For example, Section 4.4 presents a method for computing the convex hull of the mapped shape. This method's correctness is not affected by any non-border cell's sensor incorrectly reporting to not sense the shape.

Assumption 5 *Cells have memory and can perform tasks upon receiving messages.*

Having memory and simple processing abilities allows cells to accumulate data. It should be kept in mind that his memory may be a small, fixed amount, and that the processing, if even performed by a CPU, may be done using a low-speed CPU. Low power consumption, small device

size, and low cost are often essential in small nodes. For example, Berkeley's Mica Mote, pictured in Figure 3.1, uses a 4MHz processor with 4kB of ram and 128kB for program storage [18]. A method whose cell memory requirements are constant or sub-linear as a function of the number of cells is strongly desired given such resource-limited hardware. Once the data is available to a traditional computer, more resource intensive algorithms may be employed to take advantage of the gathered data.

Additionally, cell processor speeds may vary. Mass manufactured systems often vary from production run to production run, and even individual item to item. It is assumed cells cannot depend on closely-related timings.

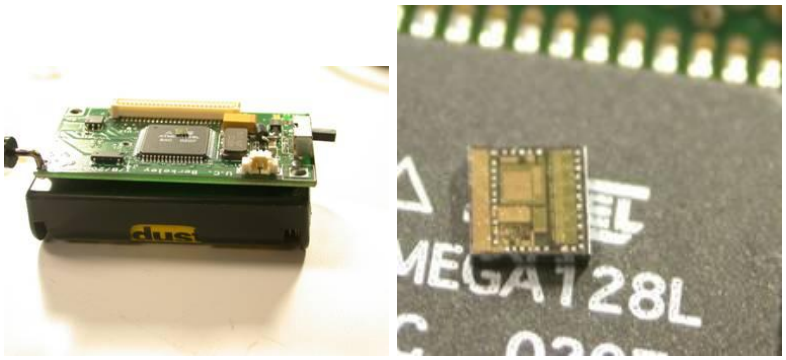

Figure 3.1: Berkeley's Mica Mote
On the left, Berkeley's Mica Mote. On the right, the newer Spec on top of the Mica's cpu. Photos reproduced from JLH Labs: `http://www.jlhlabs.com/jhill_cs/spec/`.

Assumption 6 *Cells have at least low-bandwidth communication with nearby cells.*

Global communication, as discussed in Section 3.1, requires significantly greater processor or communication abilities. The assumption that short range communication is feasible stems from short-range communication being present among entities in nature, from bird chirpings to chemical gradients inside cells. Additionally, other projects, such as Berkeley's Motes and MIT's Amorphous Computing, use local communication for the same reason. Positioning using beacons or a GPS system similarly likely has too significant processing and hardware sophistication requirements, so cells must be able to determine their physical relations solely through local interactions.

Reconstructing the connectivity of cells only requires messaging in the developed method. Reconstructing the geometrical shape may also make use of an approximate distance and

direction from the sender, though this is not required. As Lawrence notes, cell polarity plays a critical role in organism development, providing biological support for this assumption [23]. It is assumed that message sensors can determine into which of k ranges a message's distance and direction fall into. This information is then used by a traditional computer to recompute the sender's relative location. As the number of neighbors a cell has increases, the number of range distinctions needed by a cell decreases. Distance and direction information may be collected by analyzing properties of the communication medium, such as radio signal strength. Or, other cells, sprinkled about, running a different amorphous computing program, could be used. These other cells would carry the message, keeping track of the number of hops, as described in [26, 27]. Using this method, sensors with only on and off sensing give accurate information.

Assumption 7 *Cells can fail unexpectedly. Cells either work perfectly or have failed completely.*

As noted in Section 3.1, with tens or hundreds of thousands or more cells, the probability is that some cells will fail. To help simplify this model it is assumed that a cell either works or does nothing, that it will not inject incorrect data into the amorphous computer. Failsafe designs have long been studied in engineering, though these sometimes require additional complexity, and even with failsafe cells intentionally-incorrect information could still be injected into the computer, by malicious nodes, for example. Messaging among cells is also assumed to either work or to fail completely. Building reliable communication systems on top of unreliable systems is a long-studied problem. For example, TCP/IP provides reliable messaging on top of the unreliable transport IP [19].

Assumption 8 *There exists a cell that can communicate with both other cells and a traditional computer.*

To pass on the amorphous computer's gathered data there must exist a way for the amorphous computer to transfer this data to a traditional computer. Whether only one cell with this capability is required or the predominance of the cells do is decided by the querying method used, this is discussed in Section 3.4.3. It is likely that the hardware present to communicate with other cells can also be used to communicate with a traditional computer, such as a radio link. However, were the cells to communicate using the diffusion of chemicals, a non-chemical means of communication may be needed as well.

Assumption 9 *Each cell has a unique id.*

Radios, to PDAs, to power supplies have unique serial numbers. Walmart and Target are moving to label every piece of their merchandise with an RFID tag, which transit their unique id via radio. Manufacturers have for some time been able to mass manufacture identical units with unique ids.

Assumption 10 *Cells are able to determine when mapping is complete.*

A traditional computer can only make use of information present, so cells need a way of determining whether mapping is complete or not. At the cell level this is easy, a cell just remembers when its sensor is done. Local neighbors of cells can inform each other when each cell is complete. Sending completeness data beyond the local level may be unacceptably expensive, however. In this case, an upper bound on the time needed to map the shape is likely known a priori, cells can use this knowledge to largely be correct in determining when the mapping is complete. Note, though, that a cell with incomplete information could likely be treated as a failed cell would be, and since it is not assumed that all cells will function correctly, it is okay if a small number of cells do not complete their sensing.

3.2.3 Observer

Assumption 11 *A traditional computer can communicate with at least one cell after the cells have finished sensing.*

A traditional computer needs to be able to query the gathered information from the amorphous computer. The traditional computer may simply contact cells after they are complete, or it may be required that cells move away from the shape to a region where they can be communicated with. Cell querying is presented in more detail in Section 3.4.3.

A traditional computer may also not be able to communicate with the amorphous computer while the shape is being mapped. As discussed in [13], the amorphous computer may have been chosen because the region to be mapped is too hazardous for traditional computers.

3.3 Cell Primitives

The primitive actions a cell may perform in the developed method of using an amorphous computer to map a shape follow from assumptions of the environment, cell, and observer. A cell's primitive actions are listed in Figure 3.2.

1. Send and receive messages to and from nearby cells, informing neighboring cells whether a cell is on or not on a shape and the cell's id. When receiving a message, determine in which of a fixed $k \in [0, \infty)$ ranges are the distance and direction to the message's sender.
2. Sense whether it is on or not on the shape.
3. Store messages from neighbors.
4. Read its unique id.
5. Some or the predominant number of cells can communicate with a traditional computer.

Figure 3.2: A Cell's Primitive Actions

3.4 Method

With the assumptions laid out and cell primitives stated, this section presents the developed method of mapping shapes using an amorphous computer. This task is subdivided into four:

1. Placing cells onto the region to be mapped
2. Mapping the region
3. Querying gathered information from cells
4. Analyzing the queried information

This thesis project's focus was the development and assessment of the second task, the cell model and how cells acquire information and interact with other cells to gather and represent information about the intended shape. Examples of how cells can be placed are also given, though placement is largely unrelated to the amorphous computing aspect of the overall task. Querying cells is also discussed briefly, cell querying has already received considerable research, through the study of sensor works and especially sensor network databases. A traditional computer is

assumed to analyze the queried information, examples are given in the analysis of the developed mapping method in Chapter 4.

3.4.1 Placing Cells

A number of techniques exist which could be used to transport cells to the region to be explored. The cells could be placed by machine or dropped from a plane. A machine would be more useful in covering small areas, whereas dropping cells would be helpful when the region to be mapped is too hazardous to move closer to. Analogous to the healing of wounds, cells could instead be placed around the region containing the shape and could grow inwards [16, page 714]. Nagpal et. al. have studied amorphous computers making use of cell growth [25]. With the cells placed in the region containing the sought for shape, we discuss the algorithm ran on each cell.

3.4.2 Mapping the Region

The cell program ran on each cell, Cell-Shape-Mapping-Program, is given in Figure 3.3. Source to the cell program used for simulation is contained in the file `shape_cell.cpp` in the source available at [12]. Each cell runs the cell program independently of other cells. Each cell's required amount of storage is a linear function of the number of neighbors. The expected number of neighbors for cells is a logarithmic function of the number of cells to be used, as shown and further analyzed in Section 4.5.

3.4.3 Querying Cells

Cells could aggregate all their information to one cell, which would communicate with a traditional computer, but Assumptions 5 and 7 mean this approach is likely too expensive in resources, needing to scale with the number of cells used, and is dependent on a sole cell. This information could be stored in a number of cells, but this only adds to the required communication. Instead of aggregating information and then reporting it, the traditional computer could query the network as a whole, as research in sensor networks has studied. However, the added communication may again be unacceptable for lower power systems.

A traditional computer could instead query most or all of the cells directly. This approach modifies Assumption 8, cells must generally be able to directly communicate with the traditional computer. However, as discussed in Assumption 9, mass manufactured products are beginning to

```
INITIALIZE()
 1  haveSampledEnv ← false

CELL-SHAPE-MAPPING-PROGRAM()
 1  if haveSampledEnv = false
 2     then SEND-MESSAGE(id, ON-SHAPE())
 3          haveSampledEnv ← true
 4  newMessages ← GET-NEW-MESSAGES()
 5  if newMessages ≠ NIL
 6     then PROCESS-NEW-MESSAGES(newMessages)
 7          stepsSinceLastMessage ← 0
 8     else INCREMENT(stepsSinceLastMessage)
 9  if stepsSinceLastMessage ≥ stepsToWait
10     then COMMUNICATE-WITH-COMPUTER()
11     else CELL-SHAPE-MAPPING-PROGRAM()

PROCESS-NEW-MESSAGES(newMessages)
1  n ← length[newMessages]
2  for i ← 1 to n
3  do msg ← newMessages[i]
4     id ← GET-ID(msg)
5     onShape ← GET-ON-SHAPE(msg)
6     dir ← GET-DIRECTION(msg)
7     dist ← GET-DISTANCE(msg)
8     APPEND(neighborMessages, <id, onShape, dir, dist>)
```

Figure 3.3: Shape Mapping Cell Program
The cell program ran by amorphous computing cells to map a shape's area.

have this ability, such as with RFID tags. Additionally, not all cells need to have this ability because location information about a cell is independently stored by several cells. This requirement is stated in the listing of cells' primitive actions, Figure 3.2, and used in the simulation of the developed method in Chapter 4. To be queried by a traditional computer, a cell must be reachable. It may be the case that cells can be queried at the location of the shape, but in at least some uses of shape mapping, environments are hazardous, so cells must be transported to the computer. Mimicking slime molds, upon receiving a signal, cells could aggregate and travel together until they find an acceptable location, where a traditional computer would query the cells [36, page 260].

Chapter 4

Analysis and Discussion

In Chapter 3 the assumptions, primitive cell actions, and developed amorphous shape mapping method were presented and discussed. Chapter 3 also showed assumptions to be important in developing and selecting a method. However, they say nothing about models with the *same* assumptions and do not help one to make tradeoff decisions for particular assumptions in light of costs and goals. Additionally, while intuition from a method's steps can help in assessing a method, intuition can also be unknowingly misleading or incorrect. The most useful assessment of a method is based on the same situations one is planning on putting the method to use for. This chapter thus analyzes the developed amorphous shape mapping method with regard to the three types of identified descriptions, describing the methods employed to assess the shape mapping method under varying conditions, the relevance of the chosen methods of analysis to the use of the developed shape mapping method, the results, and discussion of these results. Analytical requirements on an amorphous shape mapping computer's parameters are also derived.

4.1 Simulation Environment

A simulator was developed to simulate cells running the algorithm given in Figure 3.3 and the cells' environment. Given a collection of cells and the region to be mapped (described as a bitmap graphic), the simulator distributes the nodes throughout the region with uniform randomness. Each cell then runs asynchronously in its static position on top of the given region, communicating with other cells through the simulator's provided messaging system which allows cells to communicate with their neighbors. The simulator transparently executes and handles context switching among the cells, executing them in random order. When cells have completed

running, a traditional computer queries each of the cells to ascertain whether or not they were on the shape and for the information gathered on their neighboring cells. This queried data is saved to a file, from which other programs can analyze the amorphous computer's results just as they would were the cell program ran outside of a simulator. The C++ source to the developed simulator, as well as tools developed for analysis, are available from [12].

Throughout the analysis, the developed amorphous shape mapping computer is used to map the three regions shown in Figure 4.1. The simulated cells in these analyses are fitted with *black sensors*; cells know whether underneath them is black or not. The leftmost figure, a circle, is a convex shape with large area, which should be easy to find and map. The second region contains four disconnected shapes, a circle, a rectangle, the letter π, and a shape containing a hole. The third region tests the amorphous computer's ability to describe text. Each connectivity, sample point, and polygonal description experiment, documented in Sections 4.2, 4.3, and 4.4, varies one independent variable, using 5000 cells, a cell neighborhood size of 10, a space of width 200 and height 200, 100% cells queried, distance and direction sensors with twenty levels of resolution, and is tested with each of the three regions. Although all cells used the same resolution sensors, this is not required of the developed mapping method or analysis algorithms. As can be calculated from Section 4.5, the expected number of cells in a given cell's neighborhood, $E[n_{\text{nghbrs}}]$, is thus $\frac{25}{2}\pi \approx 40$. The three variables tested analyzed how the accuracy varied with the number of sensor resolutions, the expected number of cells in a cell's neighborhood, and the percent cells queried.

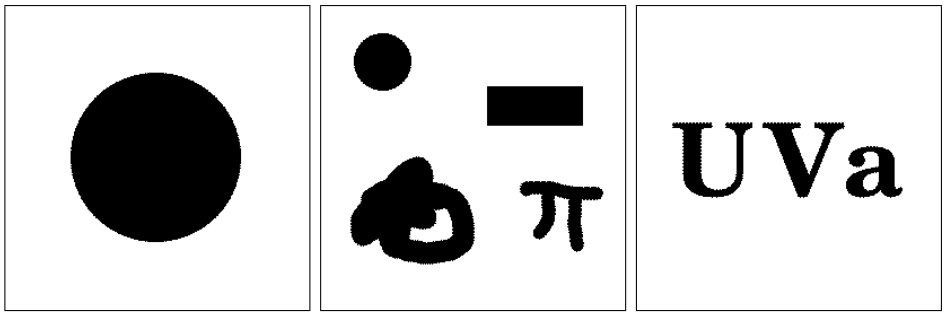

Figure 4.1: Regions and Shapes Used in Analysis
The regions used for analysis, from left to right, *circle*, *four*, and *UVa*.

4.2 Connectivity Description

An obvious application making use of knowing the connectivity of shapes in a region is determining the number of shapes present, determined by counting the number of disconnected subgraphs in the undirected graph. Intuitively, an undirected graph is a collection of objects with connections between them. A disconnected subgraph is a collection of the objects in this undirected graph, which have only connections among themselves, there exist no connections from these objects to other objects in the graph. Rosen provides a broad introduction to graph theory in [31, Chapter 7]. Building connectivity information only requires cells to be able to message their neighbors, Cell Primitive 1's (page 15) requirement for the ability to determine the distance and direction to the sender of a message is not needed. Removing this requirement allows for the use of much simpler, and thus less costly and smaller, cells.

Surprisingly, it was found that even cells without the ability to determine the distance or direction to message senders, thus only making use of cell connectivity information, provide enough knowledge for shapes to be reconstructed using existing graph visualization tools. The program `neato`, part of the `Graphviz` suite from AT&T Research, was used in this analysis to generate graph visualizations [22]. Further, this approach to shape reconstruction was found to produce more accurate mappings than cell-location methods do when the number of distinguishable ranges is small. The produced connectivity graph is independent from the physical size of the region mapped, but, cells likely have known communication distance limits, and so scale information can be obtained as well if useful.

4.2.1 Method

In addition to the queried connectivity data allowing a traditional computer to determine the topology of cells and number of shapes in the mapped region, existing graph visualization software, especially useful to circuit layout such as Very Large Scale Integration (VLSI) and Printed Circuit Board (PCB), was given the cell connection information to produce graphical visualizations. The program `neato`, part of AT&T Research's Graphviz suite, which processes undirected graphs, was used for visualization generation [14, 22]. The visualizations produced using `neato` were created using the queried cell-cell connectivity information for on-shape cells, cells not on areas being mapped were not included because of the added computational

complexity.

4.2.2 Results

The output from `neato` for the data queried from the amorphous computer for the three shapes are shown in Figure 4.2. Note that cells not on shapes were not used for shape reconstruction, thus the reconstructed shapes' orientations and relative locations are incorrect. The recomputed shapes resemble the mapped shapes even down to the cell level, aberrations are even largely preserved. The *circle* and *four* shapes are more accurately reproduced than *UVa*. *UVa*'s longer, slender pieces, as found in the U and V are troublesome. The letter a is more accurately reproduced. The amorphous shape computer's results were only analyzed for the tests shown in Figure 4.2 because *neato*'s runtime was too significant, taking as long as two days for a 5,000 cell deployment, to process results as was done for the point sampling and polygonal description tests.

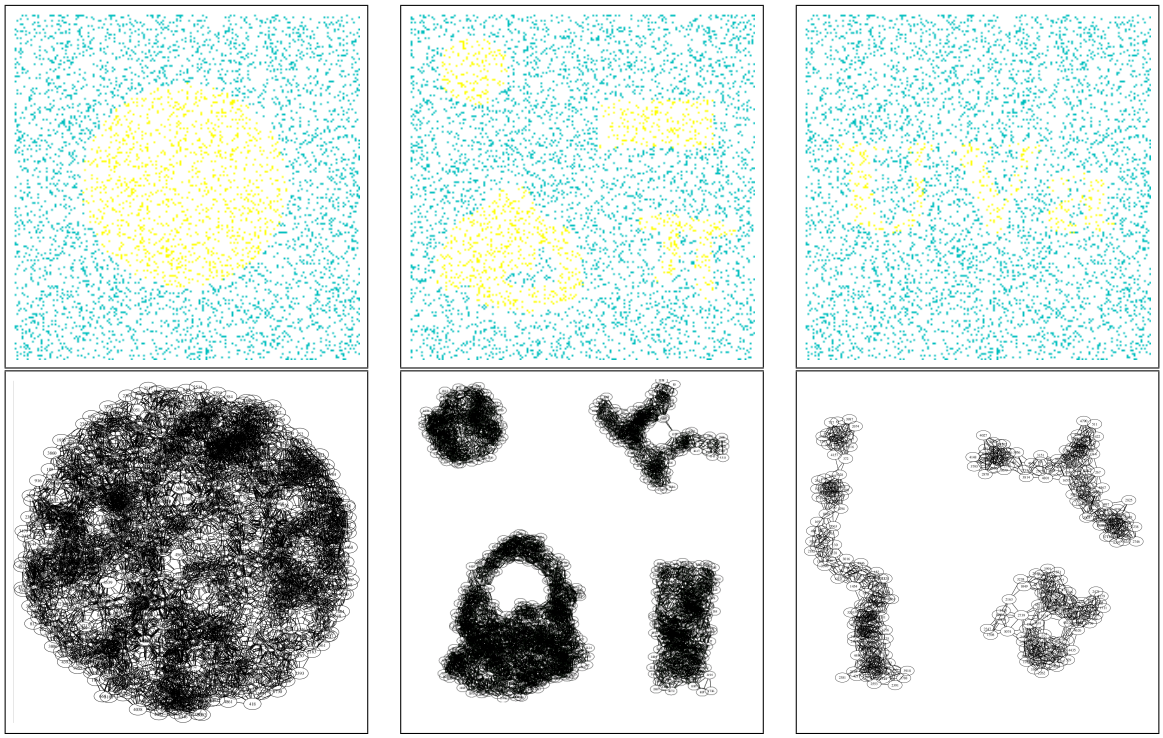

Figure 4.2: Connectivity Graph Descriptions
Shape reconstruction using only connectivity information for the shapes *circle*, *four*, and *UVa*. The top row, the actual positions of cells, the bottom row, the shapes' reconstructions.

When a subgraph becomes disconnected from the rest of the cells it is not possible for the amorphous computer to determine the relationship between the two subgraphs. The existence of

disconnected subgraphs is not the important factor, but rather, the number of cells not connected with the largest connected subgraph and their distribution in the region is. For example, were a random 10% of the cells not connected, much less information about the shape would likely be lost than if the same number of cells, which were clustered together, were lost. As the processing required by *neato* was too lengthy to do significant experiments, disconnected subgraphs were not useful to analyze for the connectivity description. However, the effects of disconnected subgraphs was analyzed for the sample point and polygonal bound descriptions in the following two sections.

Higher level software, with further knowledge of the region's characteristics, may be able to reconnect disconnected subgraphs. For example, knowing a square is being mapped and finding the amorphous computer to have mapped a small square and a larger square with a small square chunk removed, it would be likely that the smaller piece belongs in the larger piece's hole.

It should be noted that a cell failing before sending its initial message directed by the cell program (Figure 3.3) affects the results queried from the amorphous computer differently than a node failing to relay its stored information to the traditional computer. In the latter case, other cells have stored information about the failed cell; whereas in the former, no cells have any knowledge of the cell's existence.

4.3 Sample Point Description

While cell connectivity information can be used to reconstruct the mapped shape, the amount of computational work for reconstruction may be infeasible for large numbers of nodes. More importantly, making use of a cell's ability to record relative distance and direction information for message senders adds a second dimension, which can be used in shape reconstruction, allowing the creation of more accurate shape representations.

Describing shapes in terms of sampled points makes use of the direction and distance information of message senders, stored by cells to reconstruct a cell's physical relationship to the whole. A greedy, breadth first search algorithm was developed which uses cells' relative locations to give them global coordinates. Describing the shape as a collection of points transforms the problem of shape description such that computer graphics, which often deals with point representations, can be pulled into use, moving the problem from the newer, and less developed, area of amorphous computing to computer graphics, having several decades of research to build

upon. Existing pattern matching software could be used to identify the mapped shape. Another interesting use comes from Ramani et. al.'s recent work on shape searching systems [20]: by mapping a large area with the sprinkled components of an amorphous computer, objects could be later queried for when trying to locate them.

4.3.1 Method

The greedy, breadth first search algorithm in Figure 4.3 was developed to give cells global coordinates using the knowledge they gathered about neighboring cells. Given an initial cell to place, it marks this cell as having the zero coordinate and then does a breadth first search for neighbors. When a cell is first encountered, it is added to a first-in-first-out queue which is drawn from to place cells with respect to their referring neighbor. As cells have no global orientation, cell orientations are aligned as the cells are placed in the global coordinate system.

This algorithm was used because its implementation was not overly involved and its low time complexity allowed for more experimentation than another algorithm might, but note that this algorithm does not take advantage of multiple cells having knowledge of a cell when placing cells; rather, only one cell's knowledge is used. This algorithm is implemented as the function `generate_cell_shape_cov()` in the file `shape_cell_analyzer.cpp`, the implementation file for the analysis program which reads the queried cell data. An analogous depth first search algorithm was also tested. As placing cells using the fewest number of hops vs a long chain of hops would lead one to expect, the breadth first search algorithm is significantly more accurate. Under the tested conditions, more than one order of magnitude more accurate.

Cells were then given an artificial coloring, corresponding to the cell being on a shape and hearing cells which reported not being on the shape (meaning this cell is a perimeter cell), a cell being on a shape, or a cell not being on a shape. The colors given were black, yellow, and blue, respectively. An image using the cell's actual locations, which the simulator knew, was also colored to compare with the image generated from the cell's knowledge. These latter, simulator-placing based images, were not colored specially for perimeterness, only the colors yellow and blue were used. The results for each variable were analyzed qualitatively and quantitatively, by assessing the produced images and by calculating the number of cells correctly placed in space inside or outside of the shape by Figure 4.3's algorithm. Note that randomly placing cells and marking them as being inside or outside of the shape will produce cells correctly

```
PLACE-CELLS(queriedData, firstCellsId)
1  placedCellLocns ← <firstCellsId, NEW-LOCN(0, 0)>
2  PLACE-CELL-NGHBRS(NGHBRS(firstCellsId)[0], queriedData, placedCells, placedCellLocns)
3  return placedCellLocns

PLACE-CELL-NGHBRS(cellId, queriedData, placedCellLocns)
1  cellsToPlace ← NIL
2  for  each nghbrId in NGHBRS(cellId)
3  do APPEND(cellsToPlace, <cellId, nghbrId >)
4  while cellsToPlace ≠ NIL
5  do toPlaceReferId, toPlaceId ← REMOVE-FIRST(cellsToPlace)
6      if toPlaceId ∉ placedCellLocns
7        then PLACE-CELL(toPlaceId, placedCellLocns, toPlaceReferId)
8             for  each nghbrId in NGHBRS(toPlaceId)
9             do APPEND(cellsToPlace, <toPlaceId, nghbrId>)

PLACE-CELL(cellId, placedCellLocns, refCellId)
1  cellRelLocn, cellAngleOffset ← RELATIVE-LOCN(cellId, refCellId)
2  cellLocn ← cellRelLocn + LOCN(refCellId)
3  APPEND(placedCellLocns, <cellId, cellLocn, cellAngleOffset>)

RELATIVE-LOCN(toPlaceId, referId)
1  toPlaceAngleOffset ← ORIENT-CELL-WITH-REFER(toPlaceId, referId)
2  distance ← DISTANCE(toPlaceId, referId)
3  angle ← ORIENTED-ANGLE(referId, toPlaceId, toPlaceAngleOffset)
4  return D&A-TO-REL-LOCN(distance, angle), toPlaceAngleOffset

ORIENT-CELL-WITH-REFER(toOrientId, referId)
1  toOrientActualAngle ← ORIENTED-ANGLE(referId, toOrientId, ANGLE-OFFSET(referId)) − π
2  toOrientAngleOffset ← toOrientActualAngle − MEASURED-ANGLE(toOrientId, referId)
3  return toOrientAngleOffset

D&A-TO-REL-LOCN(distance, angle)
1  relX ← distance × sin angle
2  relY ← distance × cos angle
3  return NEW-LOCN(relX, relY)

ORIENTED-ANGLE(aId, bId, aAngleOffset)
1  measuredAngle ← MEASURED-ANGLE(aId, bId)
2  actualAngle ← (measuredAngle + aAngleOffset) mod 2π
3  return actualAngle
```

Figure 4.3: Cell Placement Algorithm
The algorithm used to generate cells' global coordinates using their knowledge of message senders' origins.

labeled as inside or outside of the shape 50% of the time.

4.3.2 Results

Number of Distinguishable Ranges in Message Sensing

Each round of testing was performed on cells having a different number of message distance- and direction-sensing resolutions, tested with resolutions of 1–100. The produced images are shown in Figure 4.4 and the graph of percent cells correctly marked is shown in Figure 4.5. With resolutions as low as one or two, the produced images can be used to differentiate the samples. By four to nine levels of resolution shapes' descriptions are clearly identifiable. By 15 or 40 levels of resolution the produced descriptions very closely resemble the actual positioning of the cells. The accuracy of the produced description quickly rises as the resolution is increased from 1 to 15 cells, where it settles into the 95%–98% range. Using a global cell alignment, analogous to giving each cell a compass, was found to reduce the number of resolutions required to 5 to obtain similar accuracy as approximately 20 levels without global orientation.

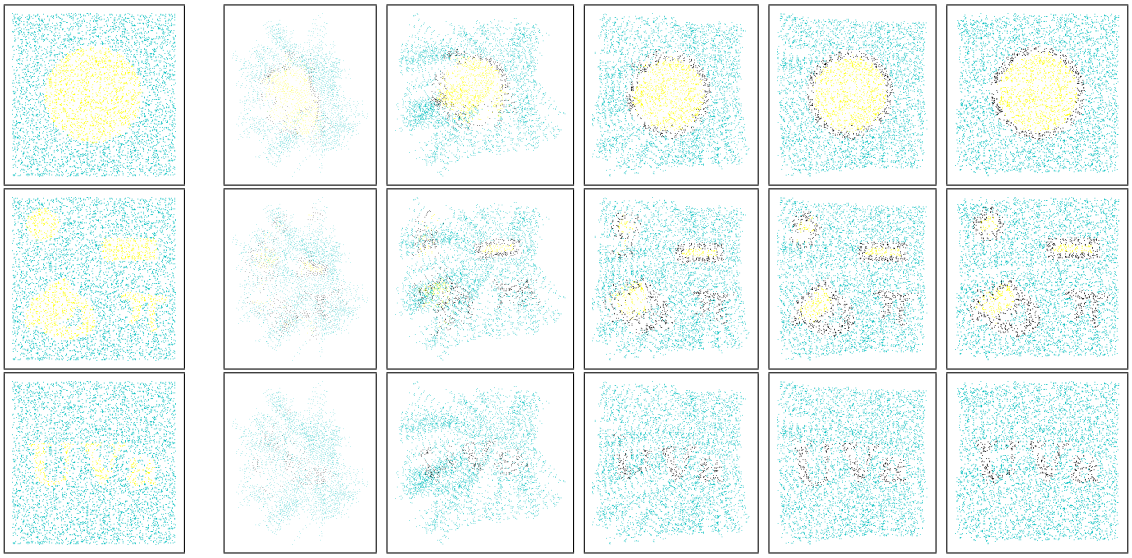

Figure 4.4: Sample Point Descriptions: Number of Sensor Resolution
Cell positioning for the shapes *circle*, *four*, and *UVa*. From left to right: actual cell positions, followed by positions using sensor resolutions of 2, 4, 9, 15, and 40.

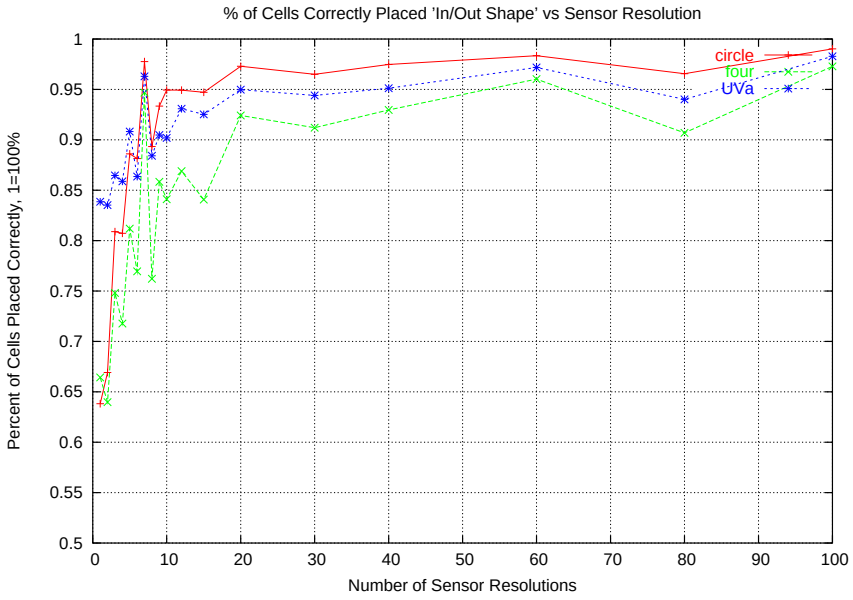

Figure 4.5: Correct Cell Placement vs Sensor Resolution

Number of Cells

Each round of testing was performed with varying total number of cells used, 500–20,000 cells. The produced images are shown in Figure 4.7 and the graph of percent cells correctly marked is shown in Figure 4.8. Figure 4.6 shows the expected number of neighbors for each of the tested total number of cells, calculated using the derived value of $E[n_{\text{nghbrs}}]$, Equation 4.1 in Section 4.5. Surprisingly, the developed placement algorithm is able to accurately place cells very soon in the shown graph, by 600 cells greater than 90% accuracy is obtained. By 1,000 cells all three shapes's reproduced images are fairly recognizable, by 5,000 they are easily recognizable. The reason for accuracy significantly increasing from 800 to 1,000 cells is that only the largest disconnected subgroup is used for description generation by the cell placement algorithm and at the low end of the number of cells placed, the number of disconnected subgraphs is significant. The distribution of disconnected subgroup sizes is shown in Figure 4.9. At 1,000 and above cells, the number of disconnected subgroups is 1, below this the largest disconnected subgroup's size vs the total number of cells quickly drops. This behavior is explained and analytically predicted in Section 4.5.

As the number of cells increases past a mid-number of cells, the accuracy of cell placement decreases. This cannot be easily seen in the reconstructed images, but the behavior can be seen in the accuracy graph by 10,000 and 20,000 cells. This is an artifact of the cell placement algorithm used, which uses a greedy approach in placing cells. As the number of cells increases, the average number of hops a given cell is from the initially placed cell increases logarithmically, increasing the accumulated error. Making use of multiple cells having relative position information about a given cell would likely lessen the loss of accuracy, it is not known whether this loss can be avoided.

Number of Cells	500	600	800	1,000	5,000	10,000	20,000
$E[n_{\text{nghbrs}}]$	3.9	4.7	6.3	7.9	39.3	78.5	157.1

Figure 4.6: Number of Cells and $E[n]$
The expected number of neighbors for a given number of total cells in a region of size 200×200.

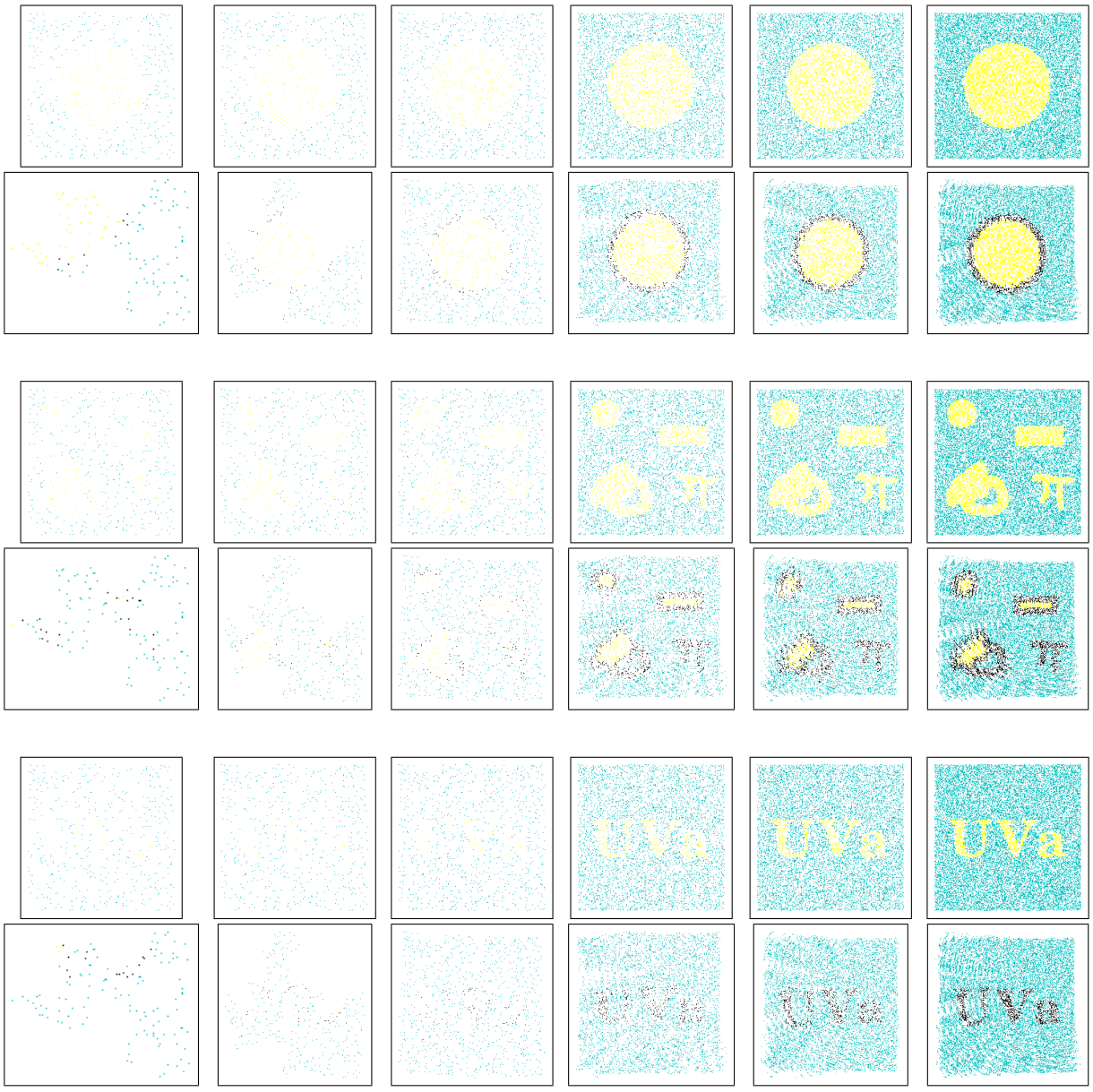

Figure 4.7: Sample Point Descriptions: Number of Cells
Cell positioning for the shapes *circle*, *four*, and *UVa*. Each two rows are the results for a shape, the first, the cells' actual positions, the second, the cells' calculated positions. From left to right, 500, 600, 1000, 5000, 10000, and 20000 cells.

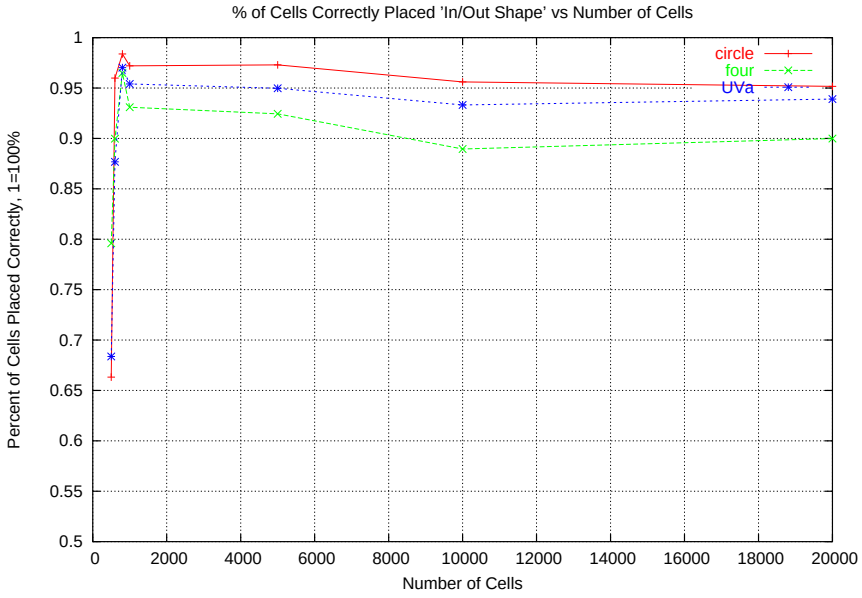

Figure 4.8: Correct Cell Placement vs Number of Cells

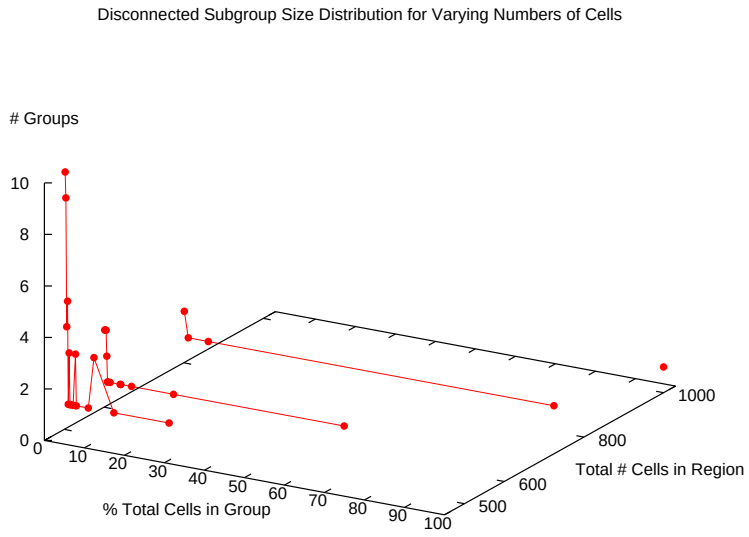

Figure 4.9: Disconnected Subgroup Size Distribution vs Number of Cells

Percent Cells Queried

Each round of testing was performed querying a different percentage of the cells, from 10%–100%. The produced images are shown in Figure 4.10 and the graph of percent cells correctly marked is shown in Figure 4.11. Varying the percentage of cells queried differs from varying the total number cells because the former does not alter the cells' interactions, only what part of them the traditional computer can query for data. The accuracy of the description rises above 90% very quickly, by 13% cells queried. Below this point, the number of disconnected subgraphs is significantly large to prevent there existing enough usable data for the used cell placement algorithm, similar to the case in Section 4.3.2. This behavior is explained and analytically predicted in Section 4.5. The distribution of disconnected subgroup sizes is shown in Figure 4.9. Once the accuracy rises, it largely stays in the $> 90\%$ range throughout the tests through 100% cells queried.

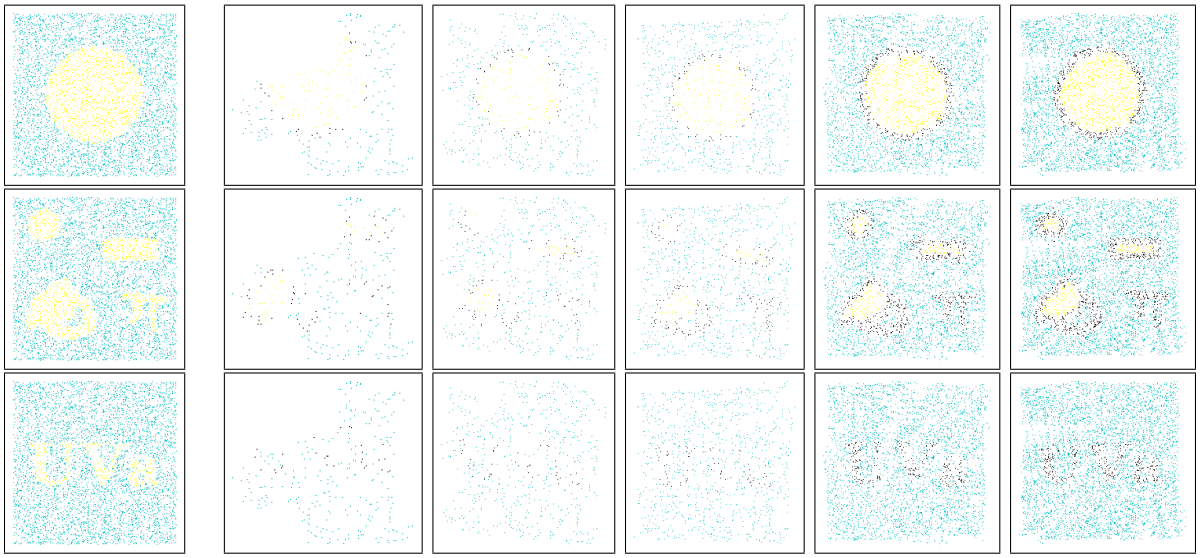

Figure 4.10: Sample Point Descriptions: Cells Queried
Cell positioning for the shapes *circle*, *four*, and *UVa*. From left to right: actual cell positions, followed by positions with percent cells queried 11, 13, 20, 75, and 100.

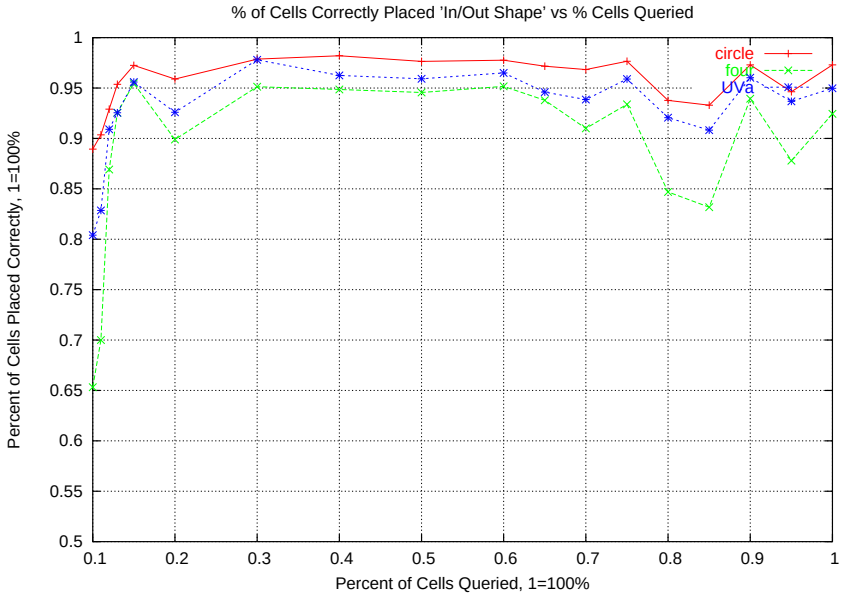

Figure 4.11: Correct Cell Placement vs Percent Cells Queried

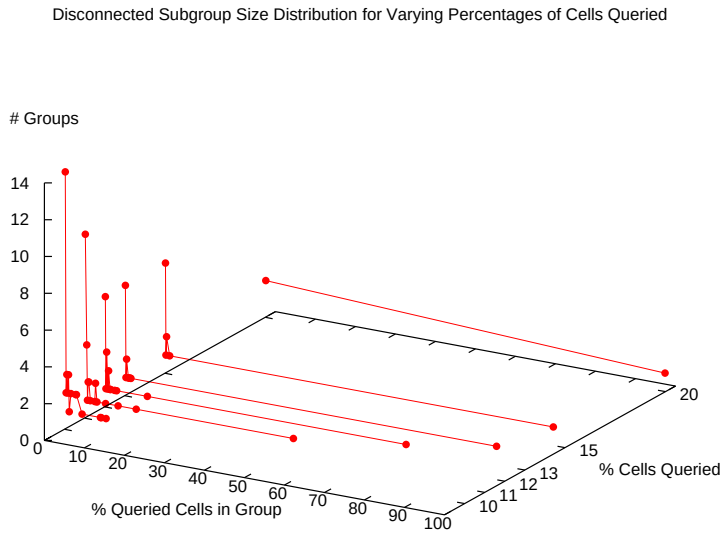

Figure 4.12: Disconnected Subgroup Size Distribution vs Percent Cells Queried

4.4 Polygonal Description

Rasterizing a shape using cells to sample the given region provides a detailed description, but a more manageable description would be more useful. For example, without knowing the bounds of the shape the question "Is this point within the shape?" cannot be answered. A bounding area also describes the sub-area of the region where the shape is located, reducing the size of the search space where the shape is located, and provides a coarse description of the mapped object's shape and size. By describing a shape using a bounding polygon existing research, especially in mathematics and computer science, can again be taken advantage of. A rectangle enclosing the cells determined by the traditional computer during the data analysis of Section 4.3.1 to be perimeter cells would provide such a bounding polygonal description. However, many other types of polygons can provide considerably more detailed information about a bounded shape. The method used in this section's analysis calculates the convex hull of the perimeter cells to describe the mapped shape.

Intuitively, the convex hull of a set of points is equivalently found by letting go of a rubber band around the given points. The shape the rubber band takes is the convex hull of the points, Figure 4.13 illustrates this. More rigorously, the convex hull of a set of points is the smallest convex set containing the points. For a set to be convex means that given any two points in the set, all points on the line connecting the two points are also in the set. Convex hulls have been studied extensively in computational geometry; in fact, the first efficient algorithm for computing the convex hull of a set began the field of computational geometry. The convex hull of a set of points is often used to describe the shape's outline, this is how it is used in this section's analysis. Preparata and Shamos give an introduction to the study of convex hulls in [29, Chapters 3–4].

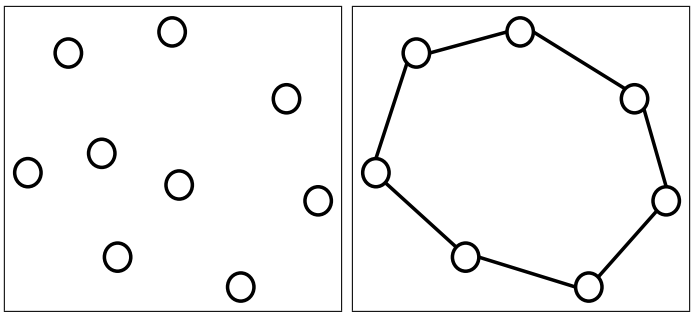

Figure 4.13: Convex Hull Example
Left, a collection of points. Right, the convex hull of these points.

Describing the mapped shape as a polygon also provides an example case of the methods used by a traditional computer affecting the assumptions made about the shape being mapped. In this case, the convex hull of a shape will not contain information about holes in the mapped shape.

4.4.1 Method

The convex hull of the cells reporting to be found, whose global locations are determined using Figure 4.3's algorithm, is computed. To assess the accuracy of each produced convex hull, random locations within the region are checked for being inside of or outside of the shape and the percentage of times the computed convex hull's answer is correct is recorded. 100,000 random locations were tested for each produced convex hull. Note that randomly placing cells and marking them as being inside or outside of the shape will produce cells correctly labeled as inside or outside of the shape 50% of the time.

4.4.2 Results

The accuracy of the computed convex hulls was found to generally mirror the accuracy of the computed point descriptions analyzed in Section 4.3. As the convex hull is computed using the perimeter points of the point description, this is expected. However, the accuracy varied much less among shapes for the convex hull description. Interesting results specific to each of the performed tests are described with the tests' analyses.

Number of Distinguishable Ranges in Message Sensing

This experiment's parameters were the same as the experiment studying the effect of sensor resolution on sample point description accuracy, analyzed in Section 4.3.2. The graph of percent sampled locations correctly marked is shown in Figure 4.14. The accuracy of the computed convex hulls varies considerably more than the point descriptions at the beginning stretch when the accuracy approaches its asymptote. The asymptote has a higher degree of accuracy than the point description, around 99% vs the point descriptions' 95%–98%.

Number of Cells

This experiment's parameters were the same as the experiment studying the effect of the number of cells on sample point description accuracy, analyzed in Section 4.3.2. The graph of

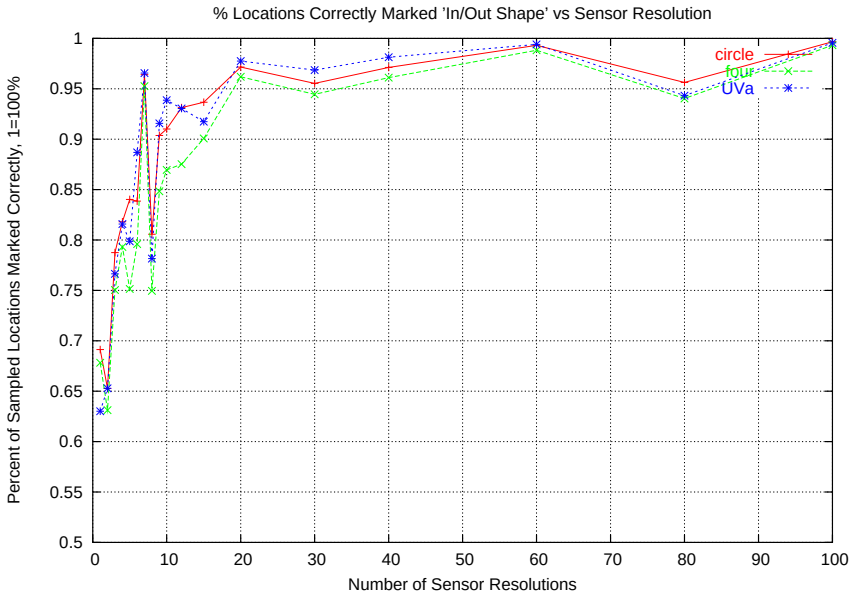

Figure 4.14: Convex Hull Shape Description: Number of Sensor Resolutions

percent sampled locations correctly marked is shown in Figure 4.15.

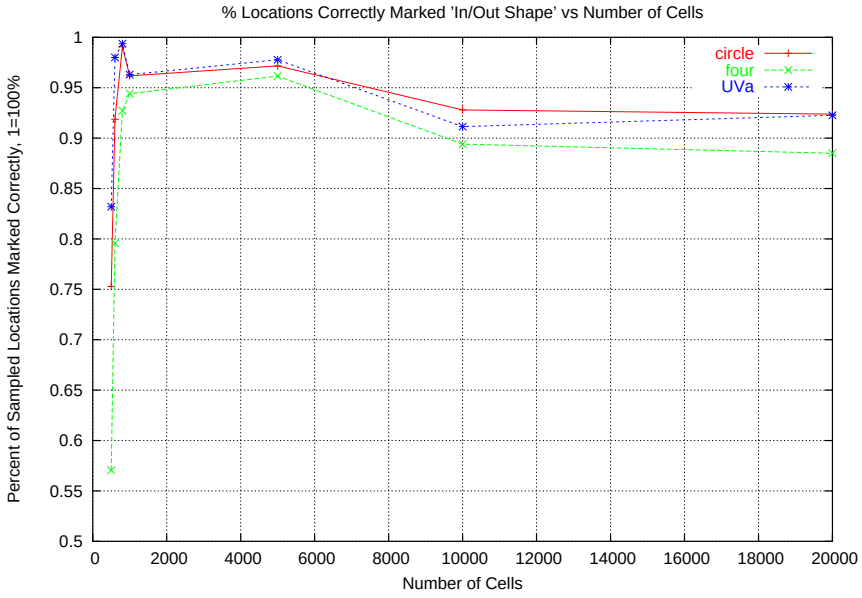

Figure 4.15: Convex Hull Shape Description: Number of Cells

Percent Cells Queried

This experiment's parameters were the same as the experiment studying the effect of the percent of cells queried on sample point description accuracy, analyzed in Section 4.3.2. The

graph of percent sampled locations correctly marked is shown in Figure 4.16.

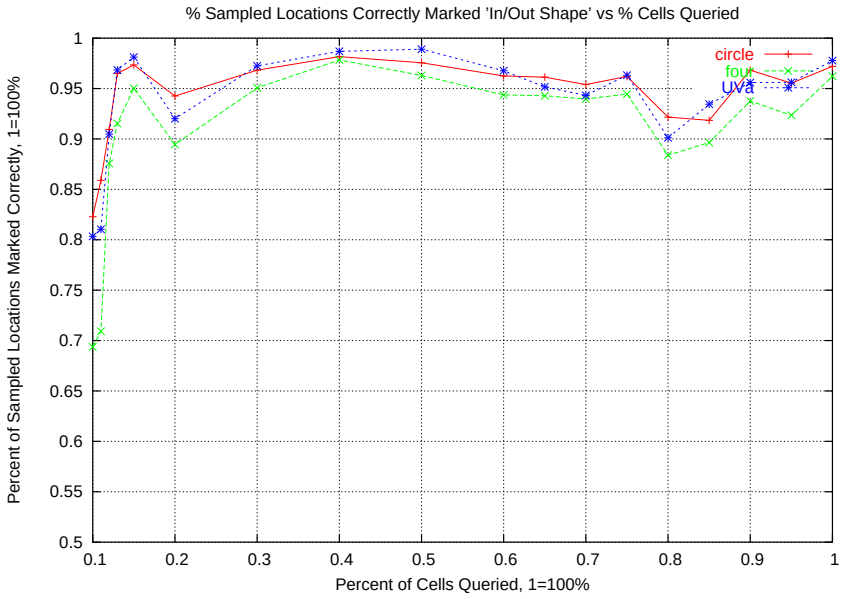

Figure 4.16: Convex Hull Shape Description: Percent Cells Queried

4.5 Largest Connected Cell Subgroup Behavior Explanations and Predictions

As noted, the experiments varying the number of neighbors, by changing the number of cells placed or the percent of placed cells queried, gained their accuracy only after passing an initial critical number of cells. Though this was not tested for the connectivity description, it was observed for both the sample point and polygonal bound descriptions, and is expected to hold true for the connectivity description as well. It was observed that below this critical point, there existed many groups of cells, each containing a small percentage of the total cells. Above the critical point, there existed one group which contained nearly all cells, and perhaps a small number of groups of a few cells. The developed method and description generation algorithms rely upon a path existing from cell i to cell j to be able to position these cells relative to each other. Thus, to be able to describe the shape of the entire region, one intuitively sees that there must be cells throughout the region, all connected in one subgroup. This agrees with the observed groupings. In setting up a deployment of an amorphous shape mapping computer, it might be helpful to know, where N is the set of deployed cells and $n_{\text{nghbrs}}(c)$ is the number of neighbors

within the given cell c's communication radius r, $P(\exists c \in N : n_{\text{nghbrs}}(c) \neq 0)$. Even more useful would be to find, where C is the largest connected subgroup, $P(|C| \geq k)$ as $k \to \infty$. Unfortunately, exact values are not known for either of these probabilities [6].

The value $P(n_{\text{nghbrs}}(c) = k)$ can, however, be derived. This is the same as there existing exactly k points in an area of size πr^2. As cells are uniformly randomly distributed through the region, the number of points in cell c's communication disc is binomial, with $p = \frac{\pi r^2}{a}$ and $n = |N|$, given that a is the area of the region being mapped. It follows that $P(n_{\text{nghbrs}}(c) = k) = \binom{n}{k} p^k (1-p)^{n-k}$, for $k \in [0, n]$. Assuming a is large compared to πr^2, this can be approximated by a Poisson distribution: $P(n_{\text{nghbrs}}(c) = k) \approx \frac{(\pi r^2 n/a)^k}{k!} e^{-\pi r^2 n/a}$. And so for our question, $k = 0$: $P(n_{\text{nghbrs}}(c) = 0) = (1 - \frac{\pi r^2}{a})^n \approx e^{-\pi r^2 n/a}$. While this may be interesting, what is really important is not the existence of a single lone cell, but the existence of a large connected subgroup. It was seen experimentally that $E[n_{\text{nghbrs}}]$ for the computer rose with the size of the largest connected subgroup, thus, knowing $E[n_{\text{nghbrs}}]$ is of some help in guaranteeing a large connected subgroup. Having determined $P(n_{\text{nghbrs}}(c) = k)$ using the binomial distribution, $E[n_{\text{nghbrs}}]$ and $\text{Var}[n_{\text{nghbrs}}]$ follow [32, Chapter 4]:

$$E[n_{\text{nghbrs}}] = np = \frac{n\pi r^2}{a} \tag{4.1}$$

$$\text{Var}[n_{\text{nghbrs}}] = np(1-p) = n\pi r^2 (1 - \pi r^2/a)/a \approx np = \frac{n\pi r^2}{a} \tag{4.2}$$

While it appears that the observed significant change in shape description accuracy is a function of $E[n_{\text{nghbrs}}]$, it does not explain why the change occurs as quickly as it does. The observed accuracy certainly does not appear to be a linear function of $E[n_{\text{nghbrs}}]$. Nor does it predict the needed $E[n_{\text{nghbrs}}]$ to be above this critical point; in deploying an amorphous shape mapping computer, it would be very useful to know a bound for this number. These and similar questions are in fact well studied, arising in many other areas, by percolation theory. Mathematicians call a large connected subgroup arising, "the emergence of a giant component," physicists call it percolation, that a phase transition occurred, sociologists say a community has been formed [3, Chapter 2]. In high school, this phenomenon arises through acid-base titrations in chemistry. A base or acid is slowly added to a container of an acid or base, when suddenly the pH level changes from acidic/basic to basic/acidic in only one drop. This section of analysis explains the observed behavior of connected cell subgroup size and provides analytical solutions for the

deployed computer's parameters to place the system above the critical point so that the entire region may be described by the amorphous shape mapping computer.

4.5.1 Background and Observed Behavior Explanation

Percolation theory studies fluid flow in random mediums. Bond percolation, where connections between points are randomly placed, is analogous this thesis project's developed method. Note that in the developed method, edges are not completely randomly placed, the distance between two cells determines their placement. However, this simplification nonetheless provides helpful insights. In determining more specific required parameters in Section 4.5.2, cell distance is also taken into account. The following explanation assumes two dimensions, though percolation often concerns the problem in general.

Rephrasing the described rapid, significant change in largest subgroup size more formally, percolation theory shows that for many types of randomly constructed systems, where the probability that an edge exists is p, there exists a critical value p_c such that when $p \leq p_c$, $P_p(|C| = \infty) = 0$ as $n \to \infty$, where C is the largest connected subgroup. And when $p > p_c$, $P_p(|C| = \infty) = 1$ as $n \to \infty$. This probability is known as the percolation probability and is one of the principal quantities in percolation, it is written $\theta(p) = P_p(|C| = \infty)$ in the literature. Then, p_c is defined as $\sup\{p : \theta(p) = 0\}$ [17, Chapter 1].

$\theta(p)$ shows there will exist a large connected subgroup when $p > p_c$, but it does not describe the subgroup's geometry. The top corner of the region, for example, may have no cells connected to C. Percolation shows this will not happen, however. Similar to the case with $\theta(p)$, the probability that the large connected subgroup contains a path from the leftmost to the rightmost point, $P_p(\mathrm{LR}(k))$, also approaches 1 as $k \to \infty$ [17, Chapter 8.8].

For this project's square lattices, it is known that $p_c = \frac{1}{2}$ [17, Chapter 11.3]. With $\theta(p)$ formally defined, the probability that there exists a cluster of infinite nodes is defined as

$$\psi(p) = \begin{cases} 0 & \text{if } \theta(p) = 0, \\ 1 & \text{if } \theta(p) = 1. \end{cases}$$

Very helpful in deploying a shape mapping computer is knowing the value of $\chi(p) = E_p[|C|]$. For $p > p_c$, $\chi(p) = \infty$; the converse is also true, for $p \leq p_c$, $\chi(p) < \infty$. This simply shows that if p

can be made to be greater than p_c, there will exist a large connected subgroup. The same behavior is true for the connectivity of two points. Making use of the FKG inequality, for the increasing random variables X, Y that $E_p[XY] \geq E_p[X]E_p[Y]$ [17, Chapter 2.2], the probability that two points x, y are connected is: $P_p(x \leftrightarrow y) \geq \theta(p)^2$. Remembering that for $p > p_c$, $\theta(p) = 1$, $P_{p>p_c}(x \leftrightarrow y) = 1$. The probability that $P_{p>p_c}(x \leftrightarrow y,$ through a finite-sized subgroup) decays exponentially as the number of nodes increases [17, Chapter 8.5].

Percolation theory also predicts the rapid change from very coarse to fine shape description accuracy [17, Chapter 1]. It is known that the subcritical phase's tail of $|C|$ decreases exponentially, $P_p(|C| \geq k) \in \Theta(e^{-k})$ [17, Chapter 6.3]. It is also known that the probability of the existence of finite-sized clusters approaches zero with exponential decay once $p > p_c$ [17, Chapter 1]. In fact, it is know that $P_p(|C| = k) \in \Theta(e^{-\sqrt{k}})$ [17, Chapter 11.4].

While all this shows there will exist a large connected subgroup covering the region being mapped when $p > p_c$, it does not show whether there could exist multiple such subgroups. Such an existence would lower accuracy of the shape description, as it would lower $E[n_{\text{nghbrs}}]$ and the relative locations of the multiple subgroups would not be determinable [17, Chapter 1]. Percolation shows that, in fact, $P_{p>p_c}(\exists$ exactly one infinite open cluster$) = 1$ [17, Chapter 8.2].

Helpful in further analysis of these described functions, it is known that that $\theta(p)$ is continuous on $(0, 1]$, that it is a nicely behaving function [17, Chapter 8.3]. It is also known that the so the described critical phenomenon exists. And, that $p_c \in (0, 1)$.

Figure 4.17 summarizes many of the above described behaviors given p's relation to p_c.

	$p \leq p_c$	$p > p_c$
$\theta(p)$	0	1
$\chi(p)$	$< \infty$	$= \infty$
$\lvert\{C : \lvert C\rvert = \infty\}\rvert$	0	1
$P_p(k \leq \lvert C\rvert < \infty)$	$\Theta(e^{-k})$	$\Theta(e^{-k})$

Figure 4.17: Graph Connectivity Behavior Summary
Summary of a graph's connectivity behavior given p's relation to the critical probability p_c.
Reproduced from [17, Table 9.1].

4.5.2 Required Parameters for Complete Cell Connectedness

The previous section explained the largest connected subgroup's size's experimentally observed behavior when varying the number of neighbors and determined the needed p to obtain

a single large connected subgraph. However, the relation between the parameters of the system and p was not described. Without knowing how to relate p to the number of cells, size of the region being mapped, and cells' communication radius, percolation is limited to explaining general behaviors and giving insight to these parameters' values. If p can be related to these parameters, however, specific expressions can be derived to aid the setup of an amorphous shape mapping computer. This exact relationship is an open problem, but Xue and Kumar in [38] prove upper and lower bounds connecting a cell's n_{nghbrs} and the value of ψ. n_{nghbrs} can then be related to the number of cells, cells' communication radius, and the region being mapped's area. Their work disproves the common belief of the existence of a magic number for n_{nghbrs} for a graph to be completely connected, such as Kleinrock and Silvester's first proposed value of six and others' proposals of similar fixed values [38].

The assumption in 4.5.1 that an edge exists randomly and independently of the existence of other edges, a Bernoulli random graph, must be made more accurate to find a useful requirement for $\theta(p) = 1$ for the developed method. In the developed method, an edge exists between a cell x and every cell y such that the Euclidean distance between x and y is less than cells' communication radius, r. Xue and Kumar showed that for the collection of cells to be asymptotically connected, that $\psi(p) = 1$, $\Theta(\log n)$ neighbors is both necessary and sufficient. Further, they proved lower and upper bounds. Letting $G(n, \phi_n)$ be the graph of n nodes with ϕ_n nearest neighbors for all cells, ζ_b be the value of ψ for $G(n, c_b \log n)$ for $b \in \{l, u\}$, and $c_l = 0.074$ and $c_u = 2/\ln(4/e) \approx 5.1774$, they thus showed: $\lim_{n\to\infty} P(\zeta_u = 1) = 1$ and $\lim_{n\to\infty} P(\zeta_l = 0) = 1$.

With $E[n_{\text{nghbrs}}]$ derived, Equation 4.1, the required parameter values to achieve asymptotic connectedness must thus be in or above the range of Equation 4.3, described by n, r, and a.

$$\frac{n\pi r^2}{a} \in [c_l \log n,\ c_u \log n] \tag{4.3}$$

The calculated bounds for $E[n_{\text{nghbrs}}]$ for each experimentally tested number of cells is shown in Figure 4.18. Note that Xue and Kumar's result is an asymptotic limit and that they did not show the n_{nghbrs} value past which it always holds, so for "small values" of n_{nghbrs} it may not hold. Nonetheless, the values of the table predict the observed behavior as closely as the bounds c_l and c_u allow. Figure 4.9 showed $|C| > 0.90\,|N|$ by $|N| = 800$, and $|C| = |N|$ for $|N| \geq 1,000$. In fact,

using ln in place of log and Xue and Kumar's conjecture that $c = 1$ agrees with this thesis project's experiments, the first experiment to contain exactly one group of all placed cells is exactly the first experiment whose $E[n_{\text{nghbrs}}] \geq \ln n$.

Number of Cells	500	600	800	1,000	5,000	10,000	20,000
Lower bound on $E[n_{\text{nghbrs}}]$	0.2	0.2	0.2	0.2	0.3	0.3	0.3
$E[n_{\text{nghbrs}}]$	3.9	4.7	6.3	7.9	39.3	78.5	157.1
Upper bound on $E[n_{\text{nghbrs}}]$	14.0	14.4	15.0	15.5	19.2	20.7	22.3
Conjectured $\ln n$	6.2	6.4	6.7	6.9	8.5	9.2	9.9

Figure 4.18: $E[n_{\text{nghbrs}}]$: Asymptotic Limit and Observed Results Comparison
Comparison of $E[n_{\text{nghbrs}}]$ between Xue and Kumar's asymptotic limit and this project's observed experimental results.

Chapter 5

Conclusions

5.1 Summary and Interpretation of Results

This thesis project developed a method for mapping two-dimensional shapes using an amorphous computer and reporting the gathered information to a traditional computer. Three types of descriptions generatable from a shape mapping amorphous computer's queried data were discussed and each description's feasibility and requirements were assessed with respect to variables affecting the cells' environment and abilities. The identified types of descriptions generatable from cells' queried data were:

1. Connectivity of shapes in the region
2. The shape in terms of sampled points in a plane
3. The shape as a polygon

The assumptions the developed method requires were detailed and discussed, and the cell's primitive actions were enumerated. It was found that accurate cell position information and shape-enclosing polygons could be produced with on-cell direction and distance sensors capable of distinguishing among fifteen levels, expected cell neighbor counts of eight, and a successful cell query rate of 13%. The required number of neighbors to ensure description of the complete region being mapped was derived. Further, it was found that accurate shape descriptions can be built from cells having no distance or direction sensors, using only cells' knowledge of local neighbor existence. Shape reconstruction based solely on cell connectivity was found to be more accurate

than reconstruction additionally using distance and direction information in cases where the number of distinguishable ranges drops below a half-dozen.

The primary reason for the observed robustness stems from groups of cells having knowledge about a particular cell and not relying on central or hierarchical control during the shape mapping phase. The developed method's memory and computation requirements are a logarithmic function of the total number of cells used in mapping and linear in the number of local cells. Redundantly storing information about local environments allows robust operation without cell complexity being dependent on the scale cell deployment. Experimental results also indicated that the accuracy of the developed cell placement algorithm making use of directional and distance information degrades as the number of cells increases; it was not determined whether this is inherent to the approach taken for the amorphous cell program or a result of the specific cell placement algorithm used.

5.2 Social and Ethical Context

A goal of this thesis project was to further the area of amorphous computing; specifically, to further our knowledge of how to design systems that are more resilient to failures and how to accomplish computations in a much more distributed manner than we think about today. This direction of research may lead us to new models of fault tolerance; engineering of biological systems, perhaps complex organisms; new environmental issues resulting from hundreds of thousands of biological cells or robots assembling each other and consuming and giving off energy; and more. As ideas such as these are developed, we should contemplate how they will affect our world and social environments. Amorphous shape mapping computers could be used to map hazardous areas traditionally too toxic for even machines or as a safer alternative to X-Ray usage, combining with nanotechnology to seek out regions and report back. Amorphous shape mapping computers could also be used to gather reconnaissance information to attack an innocent nation or group.

These applications of amorphous mapping cannot be accomplished with today's technologies. When are we likely to see benefits such as these? When are problems likely to arise? Some of these are being found now, as we begin to develop models of amorphous and swarm computing. Development of devices based upon such ideas is beginning (e.g. microelectromechanical systems

devices, nanotechnology, and bio-engineering) and is cited by some respected players as coming about in ten to twenty years [2].

5.3 Recommendations for Future Research

An improved algorithm over the shape reconstruction algorithm given in Figure 4.3, taking advantage of multiple cells having relative location information for a particular cell, could lead to significant increases in the observed accuracies. `neato` does this, and, without relative location information other than connectivity information, is able to outperform the developed shape reconstruction algorithm when the number of distinguishable ranges drops below around a half dozen. However, the amount of computation required was found to be prohibitive for deployments over 5,000 cells when deployments in the hundreds of thousands were the goal.

Exploring the dropping of Assumption 1, that shapes are static, may lead to interesting results and abilities. Moving wild life, objects in bodies of water or space may require such abilities to be mapped using an amorphous computer.

While the developed method is robust to cell failure, it assumes cells either work or completely fail, Assumption 7. Being able to make use of partially failed cells is another unexplored area. Related, it is assumed that no cells intentionally inject false information. Competitive applications might be significantly more useful if guarantees of lack of tampering or attacks can be provided.

Much of the analysis performed for this thesis project was experimental, making use of the developed simulator. This approach can be helpful in initially studying a problem, but analytical models of performance would provide more general insights.

References

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

[2] Harold Abelson, Don Allen, Daniel Coore, Christopher Hanson, George Homsy, Thomas Knight Jr, Radhika Nagpal, Erik Rauch, Gerald Sussman, and Ron Weiss. Amorphous Computing white paper. Memo 1665, MIT AI Lab, August 1999. http://www.swiss.ai.mit.edu/projects/amorphous/.

[3] Albert-László Barabási. *Linked*. Plume, New York, May 2003.

[4] J. Bard. *Morphogenesis*. Cambridge University Press, U.K., 1990.

[5] Rebecca Bloom, Catherine Chang, and Attila Kondacs. Compilation and Biologically-Inspired Self-Assembly of Two-Diemnsional Shapes. `http://www.swiss.ai.mit.edu/projects/amorphous/6.978/final-papers/attila-catie-rebecca-final.pdf`, December 2002.

[6] Almut Burchard. University of Virginia. Personal communications, May 2004.

[7] D. Coore. Establishing a coordinate system on an amorphous computer. *1998 MIT Student Workshop on High Performance Computing in Science and Engineering, MIT/LCS/TR-737*, 1998.

[8] D. Coore, R. Nagpal, and R. Weiss. Paradigms for structure in an amorphous computer. AI Memo 1614, MIT Artificial Intelligence Lab, 1997.

[9] Daniel Coore. *Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer*. PhD thesis, MIT, 1999.

[10] David Evans. Programming the swarm, July 2000. National Science Foundation Career Award.

[11] David Evans, Tarek Abdelzaher, and David Brogan. Itr: A framework for environment-aware, massively distributed computing, November 2001. National Science Foundation Information Technology Research Award.

[12] Christopher Frost. Shapsim simulator. `http://www.cs.virginia.edu/~ccf7f/shapemap/`.

[13] Christopher Frost. Amorphous shape mapping: Thesis proposal. `http://www.cs.virginia.edu/~ccf7f/shapemap/`, October 2003.

[14] E. R. Gansner, E. Koutsofios, S. C. North, and K. P. Vo. A technique for drawing directed graphs. *IEEE Transacations on Software Engineering*, 19(3):214–230, 1993.

[15] Selvin George, David Evans, and Steven Marchette. A biological programming model for self-healing. ACM Workshop On Survivable and Self-Regenerative Systems, October 2003.

[16] Scott Gilbert. *Developmental Biology.* Sinauer Associates Inc Publishers, Sunderland, Massachusetts, 5 edition, 1997.

[17] Geoffrey Grimmett. *Percolation (Grundlehren Der Mathematischen Wissenschaften; 321).* Springer, Germany, 2 edition, 1999.

[18] Crossbow Technology Inc. Mica mtot300 datasheet, April 2004.

[19] Information Sciences Instiute. Rfc 793, transmission control protocol, September 1981.

[20] N. Iyer, Y. Kalyanaraman, K. Lou, S. Jayanti, and K. Ramani. Early results with a 3d engineering shape search system. International Symposium on Product Lifecycle Management (PLM 03), Indian Institute of Science, Bangalore, India, 2003.

[21] Attila Kondacs. Biologically-inspired Self-Assembly of 2D Shapes, Using Global-to-local Compilation. International Joint Conference on Artificial Intelligence, 2003.

[22] AT&T Labs. Graphviz. `http://www.research.att.com/sw/tools/graphviz/`.

[23] Peter Lawrence. *The Making of a Fly: The Genetics of Animal Design.* Blackwell Scientific Publications, Oxford, 1992.

[24] Radhika Nagpal. *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics*. PhD thesis, MIT, 2001.

[25] Radhika Nagpal, Attila Kondacs, and Catherine Chang. Programming Methodology for Biologically-Inspired Self-Assembling Systems. *AAAI Spring Symposium on Computational Synthesis*, March 2003.

[26] Radhika Nagpal, Howard Shrobe, and Jonathan Bachrach. Organizing a global coordinate system from local information on an amorphous computer. AI Memo 1666, MIT Artificial Intelligence Lab, 1999.

[27] Radhika Nagpal, Howard Shrobe, and Jonathan Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, April 2003.

[28] Ryan Newton and Jake Beal. Amorphous infrastructure for language implementation. `http://www.swiss.ai.mit.edu/projects/amorphous/6.978/final-papers/jakebeal-newton-final.pdf`, December 2002.

[29] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[30] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Mobile Computing and Networking*, pages 32–43, 2000. `http://citeseer.nj.nec.com/priyantha00cricket.html`.

[31] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. WCB/McGraw-Hill, 4 edition, 1999.

[32] Sheldon Ross. *A First Course in Probability*. Prentice Hall, New Jersey, 6 edition, 2002.

[33] Slobodan N. Simic and Shankar Sastry. A distributed algorithm for localization in random wireless networks. `http://citeseer.nj.nec.com/556794.html`.

[34] D Arcy Thompson. *On Growth and Form, abridged edition*. Cambridge University Press, U.K., 1961.

[35] L. Wolpert. *Principles of Development*. Oxford University Press, U.K., 1998.

[36] Lewis Wolpert. *Principles of Development*. Oxford University Press, New York, 1998.

[37] Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, October 2002. `http://citeseer.nj.nec.com/wood02denial.html`.

[38] Feng Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 15(3):169–181, 2004.