Leveraging Smart Building Blocks for Adaptive Autonomous Construction

Independent Study Report Presented for the Qualifying Exam of the Department of Mechanical Engineering and Applied Mechanics

Walker Gosrich

Committee Chair: Dr. Jordan Raney
Math Examiner: Dr. Celia Reina
Committee Member / Advisor: Dr. Mark Yim

Mechanical Engineering and Applied Mechanics
University of Pennsylvania
May 28, 2019
11AM to 1PM, Skirkanich Hall Room 508,

Contents

Chapter 1

Introduction

The construction industry is ripe for innovation. As global population skyrockets and urbanization accelerates, the need for construction grows across all applications, and the democratization and safety of construction becomes more important. Automation is perfectly poised to take on these challenges. Roboticists often refer to the 'three D's' that incentivize automation of a task: dirty, dull, and dangerous. All three of these apply to construction, where difficult and tedious work results in 20% of workplace injuries in the US, according to the US Department of Labor [1]. Automation could speed construction, lower costs, and enable new types of construction. Furthermore, automation could enable construction in places where humans cannot go: the sites of natural or nuclear disasters, extraterrestrial sites like the Moon and Mars, or the bottom of the ocean.

The future of autonomous construction has strong potential, and many startup companies and media outlets have capitalized on the excitement surrounding the field. They have dreamt up 3D-printed homes and

Figure 1.1: (left) Taken from [2]: an artistic rendering of a potential future of autonomous construction: a system that uses multiple quad-rotors equipped with extrusion mechanisms to distributively 3D print a bridge in unknown, uneven terrain. (right) Taken from [3] : A rendering of a futuristic 3D-printed home design.

Figure 1.2: (left) Image taken from [4]. A snapshot of Construction Robotics' SAM100 bricklaying robot at work. (right) Image credit: MIT Media Lab/Steven Keating: A robot that uses visual servoing and the MIT Media Lab's Print in Place method to 3D print a 14 meter diameter dome out of concrete.

communities, new architectural possibilities using 3D printing and robotic labor, and intricate autonomously built structures (Figure 1.1).

The current state of the field lags behind these futuristic ideas. Most autonomous construction systems at present focus on a single, tractable aspect of construction. A company called Construction Robotics has developed a machine that lays a brick wall slightly faster than a human, while humans feed the machine bricks and take care of complicated features like doorways or corners. Other companies have developed methods to autonomously lay cement walls in wire meshes. Others focus on large-scale 3D printing: Steven Keating at the MIT Media Lab 3D printed a 14 meter diameter dome (Figure 1.2).

Supporting these industry forays into autonomous construction is a rich body of research dedicated to developing inexpensive, effective, and robust autonomous construction. This research explores a wide range of applications: rigidly-defined templated assembly, bio-inspired collective construction, and even artistic dry-stacking or fiber-wrapping tasks. These systems are varied in the purpose of the structures they build, the manner in which they are controlled, and the materials with which they build. They approach a breadth of challenging problems in the field, which we will discuss in depth in Chapter 2.

In this work, we contribute to the body of research by addressing autonomous construction in unstructured environments. These environments are more difficult for robots, which thrive in structured, controlled areas such as factories and enclosed workspaces. In order to address some of the most important applications of autonomous construction, e.g. disaster response and extraterrestrial construction, the field has taken a broad array of approaches, from building with amorphous materials to modular robot systems. We present a distributed system that uses 'smart' building blocks to build adaptively.

Distributed systems are defined by the distribution of control authority, and sometimes sensing and computation, among multiple agents. They are useful in unstructured environments because they can simplify analysis and control, add redundancy, and enable parallelization. Instead of a centralized controller that

must understand the entire environment and plan around it, distribution allows agents to make decisions more locally, based on information they gather from their surroundings and learn from communicating with nearby agents. These simple actions are executed by agents individually, and can result in extremely complex behavior that would be difficult to centrally coordinate. Furthermore, the redundancy of many agents makes any individual failure less consequential, increasing robustness.

Some distributed systems use multiple robots to build with found materials or pre-fabricated building blocks. Other distributed systems, called self-reconfigurable or modular robot systems, build with the robots themselves. Other research, including ours, hybridizes the above approaches by constructing with smart building blocks. These blocks are embedded with sensing and processing, and can communicate with other blocks and with the robots that build the structure.

The use of smart building blocks in construction is appealing because they enable some of the same capabilities as modular robot systems, without the high costs that these systems entail. In construction, modular robots can identify the position they should take in the structure based on a set of local rules or a global blueprint, and move to take this position. Through this mechanism, modular robot systems can create rapidly adapting structures in highly uncertain environments, or precisely fulfill blueprints much faster than a single assembly robot could.

In simple construction applications, structures can be well defined by a function: a wall keeps things out, a pillar holds a load, and a bridge spans a gap. If a structure is defined by a function, modular robot systems can distributively assess this function, and reshape themselves to continually perform the function as the environment changes around them.

The drawback of modular robot systems is their cost. This cost comes largely from motors. Even at scale, motors remain expensive, while the cost of the simple sensing and processing devices that construction requires fall. In realistic construction applications, building blocks may need to be heavy and large. The cost of making these building blocks into fully actuated robots could be astronomical.

Smart building blocks serve as a happy medium between self-reconfigurable robots and standard multi-agent construction. They maintain some of the capabilities of modular robot systems, while decreasing costs that come from actuation. Our goal in this work is to expand the capabilities of construction with smart building blocks towards rapid adaptive construction in uncertain environments.

Previous work with smart building blocks builds structures pre-defined with a template, built by robots that use data from the smart building blocks to determine where to place additional blocks in the structure. This is an effective way to speed template-based construction, and it is the approach we build upon in this paper. However, we argue that the capabilities that smart blocks provide when used in such a system have not been fully exploited by previous research.

In this work, we simulate a bipartite system composed of smart building blocks, and simple actuator robots that build with them. We present a system that allows the smart building blocks to control construction: they use sensors and communication with neighboring blocks to gather local information, and apply algorithms we developed to determine where new blocks should be placed to expand the structure. This distributed, environment-based control system results in an emergent structure that adapts to changing and unexpected environmental conditions.

In simulation, we apply this system to the proof-of-concept problem of building a floodwall.

The contributions of this work are:

1. A system architecture that gives smart building blocks control of the structure design
2. A distributed autonomous construction system that builds environmentally adaptive, emergent structures using only local information
3. A set of local rules that results in the successful construction of environmentally adaptive floodwalls

In Chapter 2, we give a broad overview of prior work in autonomous construction. In Chapter 3, we describe the construction system we simulated, and present the results of four experiments designed to assess the system's capabilities of responsive, robust, scalable construction.

Chapter 2

Background

2.1 Autonomous Construction

Autonomous construction is a broad field in which many disciplines intersect: computer vision, path planning, control systems, and multi-agent systems, among others. We focus here on works that address methods and systems for construction itself.

Works in autonomous construction can be well characterized by three factors:

1. **Intent of construction task:** Whether the structure is an extremely simple form of environmental augmentation, a temporary function-based structure to perform one specific task, or a permanent structure to be used as shelter or for another specific purpose

2. **Capability of construction materials:** Whether the system builds with found materials, passive, regular building blocks, prepared building blocks that might have magnets or other assistive mechanisms, smart building blocks, or modular robots capable of actuation

3. **Control strategy:** Whether the system is controlled distributively using local information, controlled distributively with global information, or controlled in a centralized manner

First, we illustrate the research that has been conducted along each dimension above. In Section 2.2, we will focus on research in emergent structures or using smart building blocks that is immediately relevant to this work.

2.1.1 Intent of Construction Task

The intent of the construction task informs the type of system that is best suited for the job. Construction tasks range from rough augmentation of terrain, to building simple structures like dams or pillars, to building highly complex, specially designed structures.

Environmental augmentation is the simplest form of autonomous construction. In environmental augmentation, material is added to an environment to make it more suitable for a certain function. This could involve making rough terrain smooth, so that a wheeled vehicle can traverse a rubble field; flattening a piece of land to serve as a foundation; or building a ramp to climb a discontinuity in the environment.

This environmental augmentation work is often undertaken with very simple building materials – even amorphous materials such as foam. In [5], Napp and Nagpal develop an algorithmic framework for creating simple function-based structures with a remote-controlled robot that deposits expanding foam. In a two-dimensional cross section of their environment, they define a set of rules that capture 'navigability' of a terrain, based on robot parameters. Using this functional definition, they develop an algorithm to determine where the robot should deposit material. They test their method in a variety of terrains, building expanding foam ramps to make these terrains navigable.

In [6], Napp et al. extend this method to three dimensions, and implement it on an autonomous system, which uses a similar set of function-based navigability rules to augment an environment with small bean bags. They use a mobile robot with an arm to place these bags, augmenting the terrain to access previously inaccessible locations. In [7], this approach is applied to a multi-robot team with multi-material construction: each robot is specialized to build with a specific material, and the team must coordinate to effectively augment the environment. This allows faster construction, as some robots can place large foam pieces while others fill in the gaps with small bean bags.

Environmental augmentation is common in biological systems, and some research takes inspiration from biology: [8] presents a system for robotic 'blind bulldozing' to clear a site of rubble, copying an approach that ants use to clear a nesting site. Their robot system pushes rubble around using only force feedback, and effectively clears the desired area.

Environmental augmentation can also take more sophisticated forms, using regular building blocks and deliberate planning, rather than relying on simple environment-based rules. In [9], Tosun et al. present a system that uses a camera and advanced modular robots called the SMORES-EP modules. The system identifies two disconnected environmental features, such as two tables separated by a gap, and finds pre-fabricated building blocks in the environment to make a traversible path between the two environmental features.

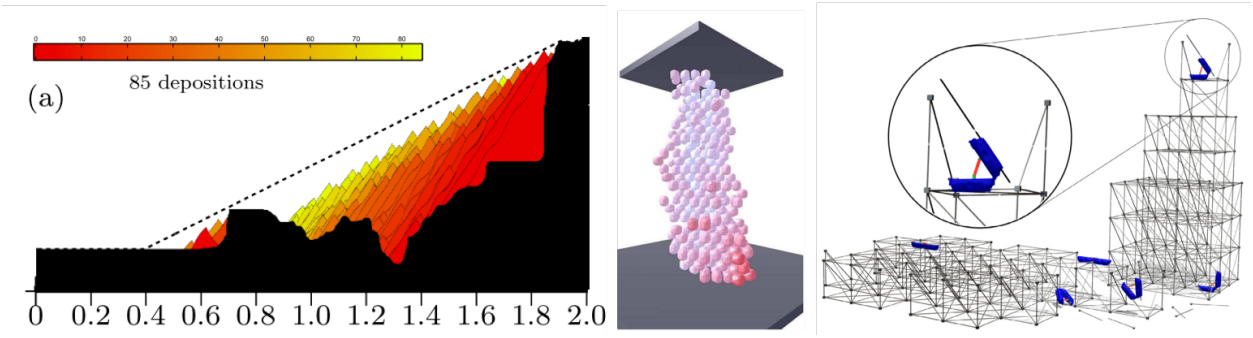

Figure 2.1: Illustrative examples of the variety of structure functions that the autonomous construction field spans. Left is environmental augmentation from [5], center is single-function structures, a pillar from [14], and right is a permanent truss structure imagined in [11].

At the other end of the construction task spectrum are permanent structures. These complex structures, such as the buildings that make up a city, are intended to be used for an extended period, and may have specific purposes that require specialized and precise features. For this reason, it is very difficult to define these structures simply. Instead, a blueprint (or *template*, as it is called in autonomous construction), is used to define the structure, and a centralized system uses one or more agents to construct this template.

Because of the nature of these structures, the systems that have been developed to build them are often sophisticated and complex. In [10], Galloway et al. present the Factory Floor system, a bipartite system that consists of floor-mounted robot arms and vertical lifting members. The system constructs a three dimensional truss by layer: constructing a layer, lifting it up, and constructing the second layer beneath it, resulting in complex truss structures from a template.

Along the same vein of truss construction, [11] presents the mechanical design of a truss system well-suited to autonomous construction, and introduces designs for a robot that can traverse these trusses. [12] works on the algorithmic side of this problem, developing an algorithm that robustly distributes construction tasks for trusses among a multi-agent system using equal-mass partitioning.

Another work that builds towards the ideal of specialized permanent structures is [13]: Lindsey et al. implement a system of multiple quadrotors capable of carrying small loads. The quadrotors work in tandem to construct templated truss structures using specialized magnetic truss members.

In the spectrum of structure intent, simple structures that can be defined by their purpose lie between environmental augmentation and complex structures. We call these structures single-function structures. They are good candidates for local, distributed construction, because their function can be distributively assessed and used to guide building without a template. Examples of these structures include walls and dams, which hold things back; pillars, which hold things up; and bridges, which span a gap.

In [15], modular robots are used in simulation to build single-function structures. Christensen et al. use

modular robots to generate simple structures that are based on functions - they create a pillar between two horizontal surfaces and a bridge that spans a gap - but do not rely on the inherent function to define the structures. Instead, they define a series of 'attraction' points in 3D space that the robots gravitate towards stochastically, resulting in a template-like control scheme.

Other works take similar approaches to building simple function-based structures: in [14], Soleymani et al. use a robot to build a wall out of bean bags. Instead of using the function-based definition of a wall, they define another type of template, albeit one that is robust and adaptive: a probability distribution, according to which building material is probabilistic deposited. They define a line of maximum probability, and center a normal distribution on this line, resulting in a wall with a normal-distribution-shaped cross section.

These works successfully build simple function-based structures, but fail to take advantage of the natural adaptability of a system that builds guided by a function. Instead of using the function that defines the structure, they define flexible templates that secondarily result in fulfillment of the structure function. Due to this reliance on templates, the systems cannot adapt to large changes in environmental conditions.

Some works in environmental augmentation use structure functions well ([5]), but beavers provide the best example: they build their dams with the goal of stopping the flow of water. If they hear the sounds of water trickling, they add more material. From this extremely simple function emerges a complex structure that adapts continuously to fulfill its purpose, and can be built by many beavers at the same time with no need for complex coordination [16].

In this work, we will take advantage of the natural adaptability of single-function structures to build like beavers do. We will use our distributed system of smart blocks to evaluate local rules (like the flowing water that beavers use) and determine where to build. In the literature, these concepts have not been applied to smart block construction.

2.1.2 Capability of Construction Materials

The material that is used for construction is another important factor that varies widely across the field. There is a rich diversity in building material: some works use materials that can immediately be gathered from the environment (found material). These materials are often so basic as to be a hindrance to effective construction, but are necessary in certain environments when more specialized building material is not available. Normal, passive building material is much easier to work with: bricks, cinder blocks, and beams are regularly shaped, and are the materials most often used in non-autonomous construction applications. Further along this spectrum is specialized building material: these materials might be *co-designed*, designed to work well with a particular robot ([17]), or have magnets to increase the area of acceptance and automatically

connect components ([13]), but they do not contain sensors or processing power. More complex still are 'smart blocks' or 'stigmergic blocks', which are embedded with sensors or processors to communicate with building robots and other blocks, and which gather information from the surrounding environment. Finally, the most complex and capable building blocks are modular robots. These combine the building blocks and the robots that build the system into one: each block is a fully actuated robot that can sense and reason about its environment.

Many works in autonomous construction make use of passive building materials. Amorphous building materials such as foams, bean bags, and adhesives are used in applications where approximate methods are implemented for highly robust structures, such as [5], [6], [7], and [14]. In [18], Napp et al. analyze the variety of amorphous materials that are used for construction, characterizing their suitability for ramp building and other tasks.

Rigid building materials are more widely used, for their strength and low cost. Simple cubic blocks are the most common, as they naturally discretize 3D space. [19] uses simple cubic building blocks in slightly less controlled terrain: it builds structures that adapt to unexpected features in the environment by incorporating them into the given template or enclosing them in a wall.

Co-designed or prepared building blocks are used in many systems that push at the boundaries of complex structure assembly in known environments. The TERMES system, developed in [17] and expanded in [20], uses co-designed building blocks that its robots can easily manipulate and climb upon. [13], described above, uses magnetic truss members with a large area of acceptance. [21] presents mechanical design of a set of building robots and co-designed cubic blocks that allow the robots to traverse the structure freely, and build in any direction.

Smart building blocks further increase structure capability; they enable distribution, storage, and collection of data about the structure and environment. They are a middle ground between expensive but capable modular robots, and unhelpful passive building blocks. In [22] and [23], Werfel et al. introduce and validate the idea of smart blocks. They implement several types of enhanced building blocks, and compare efficacy of building with these blocks. They find that smarter building blocks are effective, and result in up to an order of magnitude faster construction. In this work, we focus on extending the adaptive capabilities of construction systems that use smart blocks. In Section 2.2, we discuss the literature that uses smart building blocks in greater detail.

Finally, the most complex building block is a robot itself: modular robot systems. These systems are extremely capable, due to their distributed sensing and computation, and their parallelizability.

They are able to form structures that adapt to changing environmental conditions based on a defined function for the structure: in [24], Bojinov et al. present a modular robot system capable of this adaptive

Figure 2.2: Illustrative examples along the range of construction material capability. From left to right: found material (sticks) from Devin Carroll; co-designed building blocks in the TERMES system from [17]; 'stigmergic' smart building blocks from [26]; and the ATRON modular robot from [15].

behavior, and demonstrate a task where the modular robots form a table that redistributes its legs with changing external forces on the tabletop.

These systems are also capable of autonomously recognizing damage to built structures, and repairing it: [25] presents a modular system that recognizes and patches a hole using only local information.

2.1.3 Control Strategy

The final dimension along which we categorize autonomous construction systems is by control strategy. In distributed systems with modular robots or smart building blocks, decentralized control is often used, where individual agents distribute information or make decisions that result in global emergent behavior. In more rigidly defined, controlled environments, which often involve building more difficult structures, centralized control is often used.

The purest form of decentralized control is decentralization with no global knowledge or pre-planning: this control strategy benefits from complete scalability, and extreme adaptiveness. It relies on local control rules to result in emergent behaviors and structures. However, it is difficult to devise effective generalizable local rules, and the complexity of built structures is limited.

Systems that build with this control strategy are presented in [24], [25], and [27].

Many systems decentralize control, but use a template shared by all of the agents to decide where to build. This includes the systems presented in [17], [22], [23], [28], and [26].

Finally, systems that construct the most complex structures typically use centralized control and templates. [13] is one example of such a system, coordinating quadrotors to build complex structures. Single-robot stationary systems are another class of construction systems that use centralized control and templates.

2.2 Emergent Structures and Smart Building Blocks

In this work, we present a system architecture that adaptively builds simple function-based structures using smart building blocks. We aim for adaptiveness to changing environmental conditions, ability to respond to

environmental changes and repair damage, and scalability.

Other research has approached different components of this problem. Works such as [5] and [7] use local rules to build emergent environmental augmentation structures. However, these works rely upon centralized systems to build and maintain a representation of the workspace, and to coordinate with the agents. This limits scalability and the environments in which such a system can be applied.

Other works such as [24] and [27] use local rules that result in emergent structures - structures that are not pre-defined, but which are 'designed' as they are built through a function-based set of rules. These works use modular robots to form structures that are highly adaptive to changing environmental conditions. As discussed in Section 2.1.2, modular robot systems can be prohibitively expensive at scale. In this work, our goal is to achieve this highly adaptive behavior using less expensive smart building blocks.

In some work, smart building blocks have been used to construct template-based structures, but have not been applied to adaptive systems. In [23], Werfel et al. extend their earlier work with stigmergic blocks, validating the idea of using smart building blocks for autonomous construction. They build the same two dimensional structure with three types of building blocks: inert blocks, writeable blocks, and smart blocks. The inert blocks are simple passive building materials, the writeable blocks can hold data from passing building robots, and the smart blocks can actively communicate this information between building blocks and to the building robots. They found that by increasing complexity of building block, they could decrease building time by almost an order of magnitude.

In [28], Werfel et al. extended this approach to three dimensions, and developed in detail a method for propagating signals across the structure surface to tell the building robots where to go. We use this signal propagation method, called *gradient propagation*, in our system. We detail the method in Section 3.2.1.

In these works, Werfel and his colleagues find effective ways to construct with smart building blocks, but do not leverage the distributed sensing and processing fully: they do not make the system adaptive to environmental factors, instead building pre-defined templates. Furthermore, in [23] and [28], the authors use some non-local information in their system to make guarantees about buildability of a structure.

In this work, we present a system that is based on the system architecture of [28], with several key differences. Instead of relying on pre-defined templates, we use smart building blocks to control the structure design. This enables far greater responsiveness. We also use only local information, eliminating the small amounts of non-local information required in [28]. This increases the scalability of our system. Through these changes, we present a system that is better suited for construction in unknown and unstructured environments.

Chapter 3

Research

3.1 Research Statement

Our goal in this research is to determine whether smart building blocks can be used to build adaptive structures in unstructured environments.

In simulation, we develop a system architecture of actuator robots and smart building blocks that is capable of adaptive construction. The building blocks control construction using local rules, resulting in emergent structures that perform simple functions. We demonstrate that this system is adaptive to different environments, is capable of reacting to damage and changes in the environment, and is scalable.

In this section, we describe the system architecture and our simulation. We lay out the algorithms used by the smart building blocks to build emergent structures in Section 3.3. We demonstrate our system's adaptiveness, responsiveness, and scalability with four experiments in Section 3.4.

We choose a floodwall as a test structure for our system. The use cases for a floodwall align well with the advantages of an adaptive system that uses smart blocks. Floodwalls are:

- Built in dangerous and unstructured environments (disaster areas)
- Simple structures that can be easily defined by their function (keep water out)
- Temporary structures that must respond to changing environmental conditions (e.g. rising water levels)
- Highly susceptible to damage and failure
- Large scale

A floodwall is an effective test case, but it is far from the only application of this system. We do not focus on the specific physical implementation of our system on a floodwall, choosing instead to emphasize

Figure 3.1: From [29]: images of real floodwalls. The structure on the left took 419.8 man-hours to build. The structure on the right requires heavy machinery and human operation in a dangerous flood environment.

the fundamental questions of system functionality. Implementing this system in a physical prototype is not trivial, but it is not the focus of this work.

3.2 System Overview

We present a system composed of two parts:

1. **Smart building blocks:**

   These cubic building blocks are equipped with simple sensors and processors. They compose the structure itself, and dictate where blocks are added to the structure using local rules. They propagate gradients across the surface of the structure to guide the actuator robots.

2. **Actuator robots:**

   These robots carry building blocks across the surface of the structure, and deposit them on the structure according to the directions of the building blocks. They navigate to valid deposition sites by following the gradients propagated by the building blocks.

This section describes how each part of the system functions, and lays out the assumptions about the capabilities and limitations of each system component.

3.2.1 Smart Building Blocks

The smart building blocks form the structure itself, distributively control the expansion of the structure, and propagate gradients across its surface that guide actuation robots to deposition points.

We make the following assumptions about the building blocks:

1. They are cubic in shape

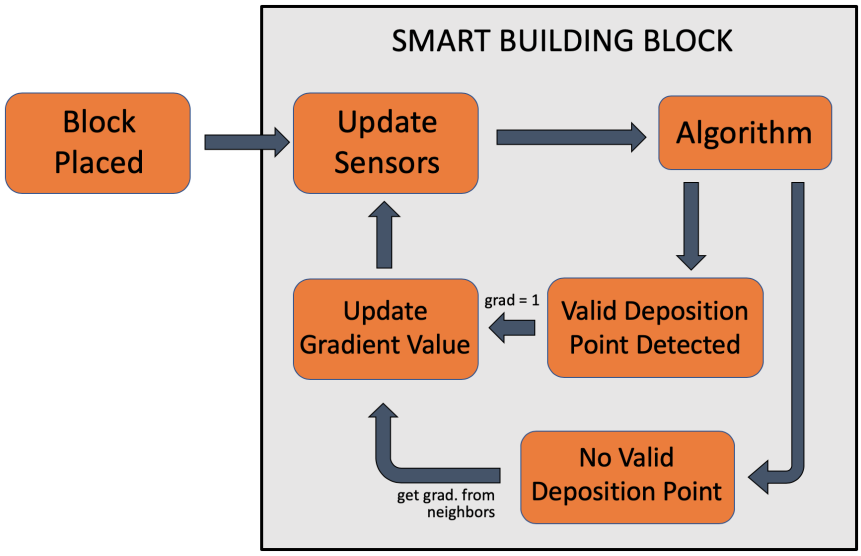

Figure 3.2: Flowchart showing the actions of the smart block in the system

2. They can rigidly connect on all 6 faces to other building blocks, and can communicate through these connections
3. They can connect to actuator robots on all 6 faces, and can communicate with the actuator robots through these connections
4. They are equipped with proximity sensors that can sense if adjacent positions are occupied (by blocks or existing parts of the environment)
5. They are equipped with water sensors that can determine the presence of water on all 6 faces
6. They are equipped with hydrostatic pressure sensors on all 6 faces
7. They have a processing unit capable of storing and processing a few small integers (on the scale of n, the number of blocks in the structure)

The smart building blocks have two functions in the system: distributive control of structure expansion through local rules, and communication of gradients across the structure surface to guide actuator robots to valid deposition sites.

Distributed Control:

Each building block is equipped with simple sensors that allow it to gather information about its immediate surroundings, and a processor that enables it to synthesize this information. Using a predefined algorithm relying on this sensor data and information collected from neighboring blocks, each block determines which

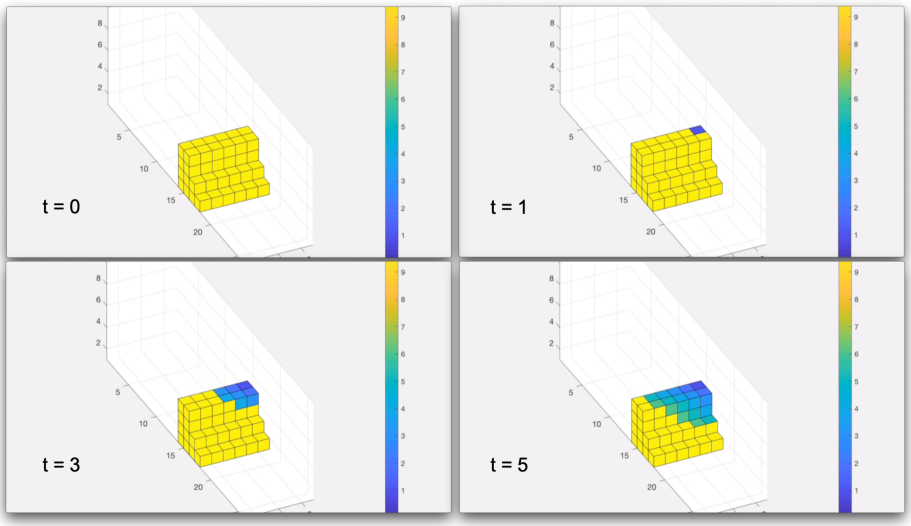

Figure 3.3: Steps in the process of a gradient propagating across the structure from a source valid deposition site. At $t = 0$, a valid deposition site is designated at the top right corner of the structure. The gradient value is represented by the color, with blue $= 1$ and yellow > 10. At each time step, the surrounding blocks check their neighbors for a smaller gradient value and increment this value. At $t = 3$, the gradient has propagated a distance of 3 blocks, and at $t = 5$, it has propagated 5 blocks in all directions.

of its 6 faces are valid deposition points for additional blocks. The algorithms that are used to make this decision are described in Section 3.3. After determining the valid deposition points that neighbor each building block, it communicates this information with neighboring blocks through *gradient propagation*.

Gradient Propagation:

The blocks propagate gradients along their surface to signal the shortest path to a deposition point to the actuator robots. This mechanism allows the actuator robots to move directly to deposition points, rather than performing a random walk or systematic search across the structure surface, or relying on a complete state representation of the system and localizing within it. This idea was originally proposed in [22] and extended to 3D blocks in [28].

In broad terms, the building blocks propagate gradients through the following steps:

1. Each block determines whether it borders a valid deposition site
2. If the block borders a deposition site, it sets the gradient value at the bordering face to 1
3. If the block does *not* border a deposition site, it updates gradient values by incrementing the lowest gradient value from its neighboring blocks

In practice, the implementation of this method is slightly more complex: each block face on the surface of the structure belongs to a building block, but must act as an individual unit. If the face borders a valid

deposition site, it receives gradient value = 1. If the face does *not* neighbor a deposition site, the block determines the set of faces that are adjacent to the face, identifies the smallest gradient value from this set, and increments this value by 1. This is the final assigned value.

But these gradient values are not directly useful to an actuator robot. When attached to a specific face, the robot only sees one gradient value, and can not determine which direction to move. For this reason, each block also stores the direction from which each face 'received' its gradient: North, East, South, West, Up, or Down. This results in a vector field across the structure surface, indicating gradient direction. An actuator robot uses these vectors to identify the direction it should move, and follows these directions until it reaches a face with gradient 1, where it knows to deposit a block.

These two mechanisms, distributed control via local rules and gradient propagation, together enable the blocks to efficiently guide construction, while using only local sensor readings and communication.

3.2.2 Actuator Robots

In a self-reconfigurable robot system, each robot is capable of a full range of motion. In order to decrease the prohibitive cost of such a system, our system removes this actuation authority from the building blocks, and concentrates it in a relatively small number of actuator robots. The actuator robots are responsible for transporting the building blocks from the depot to the deposition points, following the directions communicated by the building blocks. They are equipped with sensors, processors, and actuators that enable this task. We assume that each actuator robot:

1. Can hold one block at a time
2. Can move along the surface of the structure freely in all three dimensions, by attaching to one block at a time
3. Can deposit a block in any valid position adjacent to the robot
4. Can store and process a few small integers (on the scale of n)
5. Can communicate with the block to which it is attached
6. Given a direction of movement (North, East, South, West, Up, or Down) from a block, can determine whether it will need to move around a corner or to an adjacent block, and whether this movement is possible
7. In present implementation, we do not address the navigation of the actuator robot back to the depot from a point in the structure. This can easily be achieved with a second gradient propagated from the

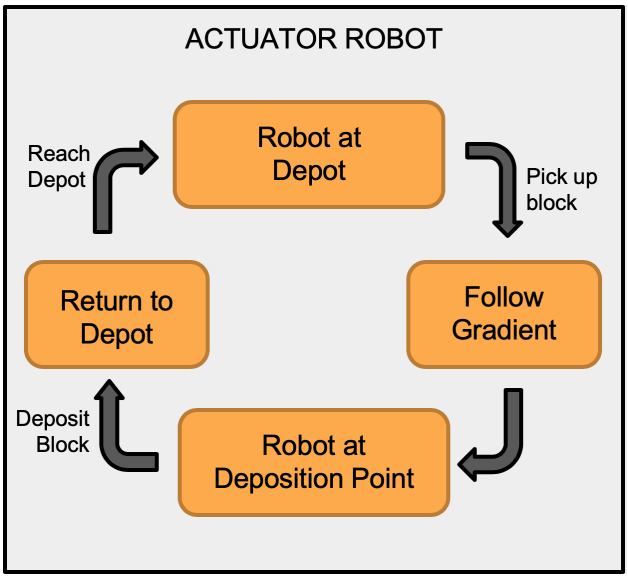

Figure 3.4: The finite state machine under which each actuator robot operates

depot, but for now we assume the actuator robot is capable of this navigation independently.

The actuator robots function under a finite state machine, depicted in Figure 3.4. An actuator robot is initialized at the depot, where it picks up a block and begins following the gradient on the structure surface. The robot follows the gradient until it reaches a block with gradient value 1: a valid deposition point. It then moves to an adjacent position, and places the block at the deposition point. It switches its state, and returns to the depot, repeating until there are no valid deposition points remaining.

3.2.3 System Initialization and Other Assumptions

The system is initialized by a single 'seed' block. This block is placed in the environment by outside forces, and for the purpose of this investigation, is identical to all other building blocks. The structure grows from this seed block based on local rules.

We assume the depot is an effectively infinite supply of smart building blocks, and we locate the depot on the surface of the structure at the x- and y-positions of the seed block.

Finally, the space in which the system functions is discretized into a 6-connected cubic grid. All computation and movement takes place in a discrete manner within this grid.

3.3 Algorithms

The algorithms presented in this section are used by the smart building blocks to determine which neighboring positions are valid deposition points. The structure emerges from these collective decisions and the

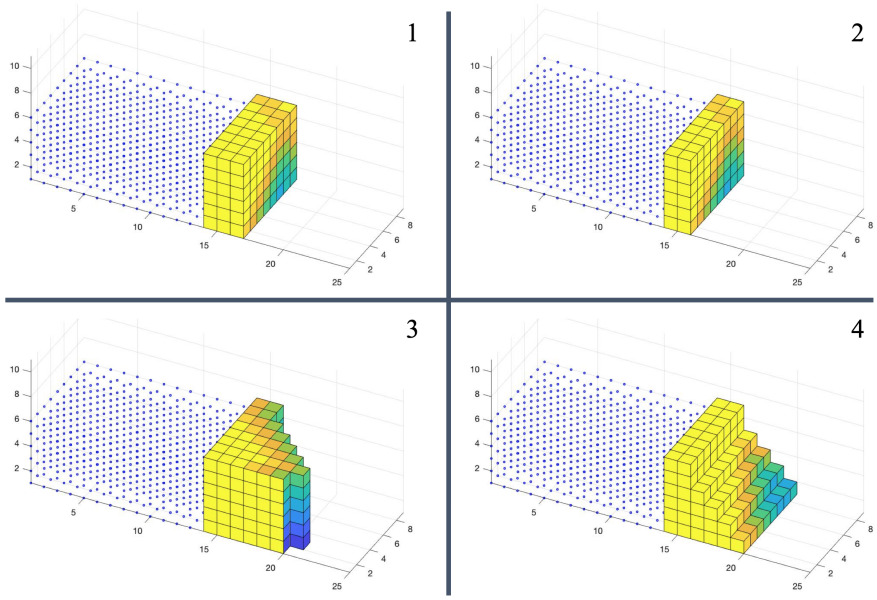

Figure 3.5: The structure that the system builds with each algorithm in an identical environment. Top left is the template algorithm, top right is the fixed-width algorithm, bottom left is the blind algorithm, and bottom right is the hydrostatic algorithm.

environmental factors that inform them. In this sense, the algorithms determine the shape and function of the final structure. The shapes that emerge from each algorithm given identical inputs are shown in Figure 3.5.

In this section, we describe the four algorithms that we tested. Each of the algorithms takes as input the readings from sensors and data about surrounding blocks that is collected through local communication. The algorithms output the directions of valid deposition sites. The smart building block uses this directional information to propagate the correct gradients, and to indicate to the actuator robot where to deposit new building material.

The algorithms vary in the complexity of their inputs and logic, and in the shape of the structures that emerge from them. The inputs and capabilities of each algorithm are summarized in Table 3.1. We present them in order of increasing complexity and capability.

3.3.1 Template

The template, shown in Algorithm 1, is the simplest possible algorithm to distributively control construction. It is not an adaptive algorithm, and it does not rely upon sensor inputs. Instead, it is a blueprint that is decided by a human operator, based on knowledge of the site in which the system is to be deployed.

Templates are used in [22] and [28] to effectively build structures in a system such as ours. The drawback

Table 3.1: Summary of inputs and capabilities of the four algorithms discussed in this section.

	Template	Blind Algorithm	Fixed-Width Algorithm	Hydrostatic Algorithm
Pre-specified parameters:	**Full template**: length, width, height	Water containment direction	Water containment direction, **width**	Water containment direction
Sensors used:	N/A	Water sensors	Water sensors	Water sensors, pressure sensors
Block ↔ Block communication:	Gradients, directions	Gradients, directions, **type gradient**	Gradients, directions, type gradient, **width gradients**	Gradients, directions, type gradient, **hydro gradients**
Adaptive to unexpected/changing environment	**N**	Y	**Partially**	Y

of this method is that it is not adaptive to changing environmental conditions: perfect knowledge of the environment is required beforehand, and this severely limits the applications of autonomous construction. In the floodwall scenario we investigate here, a template-based construction strategy would require knowledge of the exact extent of future flooding. This knowledge is not typically available. A template would also fail in the case of unexpected obstacles in the construction area or significantly uneven ground terrain.

However, the template requires very little communication between blocks. It only requires that a block receives from its neighbors its position in the discretized space, and knowledge of where its neighbors are. From here, it simply consults a set of stored parameters that define the template itself, and checks its position against these parameters.

We use the template algorithm as a ground truth for comparison of our other, adaptive algorithms. For autonomous construction with smart building blocks, it is the state of the art. We aim to improve upon the capabilities of a system that uses this algorithm, while paying only marginal cost in number of messages sent.

The template algorithm, shown in Algorithm 1, uses the user-provided template to check whether a given position is valid. Each block, upon placement, receives its position in some shared reference frame from its neighboring blocks. It is pre-loaded with the parameters that define the template. In the case of the floodwall, these parameters are the maximum and minimum extents of the floodwall in each dimension (six parameters). The algorithm iterates through six directions: North, East, West, South, Up, and Down, and finds the neighboring position for each direction. It checks this neighboring position for validity (is it

occupied, and is it within the bounds of the 'world') and checks to see whether it is an occupied position within the template. If both of these are true, the position is marked as a valid deposition site, and added to the list of attachments.

Algorithm 1: Template algorithm

```
Result: Valid attachment sites for a block
attachments ← [];
foreach face_direction do
    neighbor ← GetNeighbor(face_direction);
    within_template ← CheckTemplate(neighbor);
    is_valid ← CheckValidity(neighbor);
    if within_template & is_valid then
        attachments ← [attachments,neighbor];
    end
end
```

3.3.2 Blind Algorithm

The 'blind' algorithm, shown in Algorithm 2, is the simplest set of rules that adaptively builds a structure to contain the water. The system takes only a direction to contain the water (containment direction) as user-provided input, and builds in every direction except for this user-specified direction. We refer to the algorithm as 'blind' because it uses only water sensor data, and builds blindly wherever there is water. This results in a structure that expands uniformly from the depot, as the actuator robot always fills the closest valid deposition point, and thus is biased towards the depot. This can be seen in Figure 3.5.3, a sample structure constructed by this blind adaptive algorithm. The depot in this case is located in the near left corner of the structure, and so the actuator robot deposits the blocks nearest to this corner first.

In order to deal with the direction input from the user, the blind algorithm (and the two algorithms presented later in this section) makes use of special block-to-block communication. Each building block has a binary 'exterior' property, which indicates whether the building block is located on the surface of the structure that will be directly containing the water. The exterior property is by default false. The seed module that initiates construction is on the exterior, and passes a *true* exterior signal in all directions perpendicular to the containment direction. In this way, all blocks in the structure know if they are exterior or interior blocks from the binary signal, and in this sense, the blocks are separated into two groups that the algorithm treats slightly differently.

Unlike the template algorithm presented above, the blind algorithm is adaptive to changing environmental conditions: if the water level rises, the structure will grow to contain it. If the channel widens, the structure will grow to fill it.

The blind algorithm is detailed in Algorithm 2. If the block is an exterior block (on the surface that will contain the water), it iterates through all directions (North, East, South, West, up, down) and checks if: 1. there is water in the direction, 2. the position in the direction is a valid one (i.e. unoccupied), 3. the direction is *not* the containment direction. If all three of these checks are positive, the block designates the position in the direction a valid deposition site.

If the block is interior, it performs the same iteration, but it does not check if the direction is the same as the containment direction. Since it is on the inside, it does not need to ensure that it does not build into the water containment area.

Algorithm 2: Blind adaptive algorithm

```
Result: Valid attachment sites for a block
attachments ← [];
if block_type = exterior then
    foreach face_direction do
        neighbor ← GetNeighbor(face_direction);
        is_wet ← water sensors;
        is_valid ← CheckValidity(neighbor);
        if face_direction ≠ out & is_wet & is_valid then
            attachments ← [attachments,neighbor];
        end
    end
else
    foreach face_direction do
        neighbor ← GetNeighbor(face_direction);
        is_wet ← water sensors;
        is_valid ← CheckValidity(neighbor);
        if is_wet & is_valid then
            attachments ← [attachments,neighbor];
        end
    end
end
```

This algorithm will only stop building when all non-exterior surfaces are dry. This guarantees that, given an infinite supply of building blocks, the system will eventually contain the water. However, the structure is, by inspection, larger than necessary and asymmetrical, indicating a sub-optimal distribution of construction material.

It also means that we must make the assumption that water behind the wall will drain when the water in front of the wall is contained. If it is not drained, the wall will continue building backwards, which is not a desired behavior.

3.3.3 Fixed-Width Algorithm

The fixed-width algorithm takes a user-specified width, and builds adaptively, maintaining this width. An example of a structure built using this algorithm is shown in Figure 3.5.3. The structure pictured was built with a specified width of 2 units.

Like the blind algorithm, the fixed-width algorithm also makes use of a special block-type signal, that allows blocks to determine whether they are on the exterior or interior of the structure through only local communication.

The fixed-width algorithm uses an additional specialized signal to guarantee that it will maintain the desired structure width. This signal is similar to the gradient that is propagated across the structure surface: it counts width, and is incremented by every block as it passes the gradient on. This gradient only moves in the opposite direction from the specified containment direction. In this manner, each block is able to determine its depth from the exterior of the structure through only local communication.

Using this depth value, water sensor values, and local communication with neighboring blocks, the fixed-width algorithm determines the neighboring positions that are valid for block deposition.

The fixed-width algorithm, detailed in Algorithm 3, functions very similarly to the blind deposition algorithm. If a block is exterior, it iterates through all 6 directions, checking the water sensor values and the validity of the position in each direction. If the sensor readings and validity check are positive, and the direction is *not* the specified containment direction, the algorithm determines the position to be valid.

If the block is interior, the algorithm takes into account the structure width signal. For the direction opposite the specified containment direction, the algorithm only designates the adjacent site a valid deposition site if the width at the block is less than the specified goal width. For all other directions, the algorithm simply checks if the position is valid. Since the algorithm does not use the water sensors for interior blocks, it prohibits building upward by interior blocks. This ensures that the interior blocks only build as high as the exterior wall.

This algorithm guarantees construction of a wall that will contain water in the given containment direction, and construct to the specified wall width. Like the blind algorithm, it is adaptive to changing water levels. However, requiring the user to input a desired width assumes that the user has knowledge of the necessary width to make a stable wall. If the user only expects small water levels and inputs a correspondingly small width, but then unexpectedly high water levels result in a tall and very skinny wall, the structure is likely to fail. To avoid this, we make use of another set of sensors in the hydrostatic algorithm.

Algorithm 3: Fixed-width adaptive algorithm

```
Result: Valid attachment sites for a block
attachments ← [];
if block_type = exterior then
    foreach face_direction do
        neighbor ← GetNeighbor(face_direction);
        is_wet ← water sensors;
        is_valid ← CheckValidity(neighbor);
        if face_direction ≠ out & is_wet & is_valid then
            attachments ← [attachments,neighbor];
        end
    end
else
    foreach face_direction do
        neighbor ← GetNeighbor(face_direction);
        ok_width ← (face_direction ≠ GetOppositeDirection(out)) OR (GetNeighborWidth(out)+1
          < goalwidth);
        is_valid ← CheckValidity(neighbor);
        if ok_width & is_valid & face_direction ≠ up then
            attachments ← [attachments,neighbor];
        end
    end
end
```

3.3.4 Hydrostatic Algorithm

The hydrostatic algorithm is the most complex, and uses the most sensors. However, it requires no knowledge from the user, making it more robust and adaptive than the fixed-width algorithm. It also builds a smaller structure to fulfill the same purpose as the blind algorithm, making it more efficient.

The hydrostatic algorithm accomplishes this by making use of water sensors, hydrostatic pressure sensors, and local communication with neighboring blocks. It requires only a containment direction from the user.

It uses the same specialized local communication as the blind and fixed-width algorithms to propagate information about interior vs. exterior blocks.

The hydrostatic algorithm uses a similar method to the fixed-width algorithm to propagate information about width of the structure. The exterior blocks read the hydrostatic pressure sensors, and determine the appropriate width from this information. In our simulation, we supply these hydrostatic pressure sensors with values using Equation 3.1. The desired width is calculated from this pressure reading by simple linear scaling in Equation 3.2, with $\alpha = \frac{1}{\rho g}$, $\beta = 0$. The exterior blocks then propagate a gradient in the opposite direction of the specified containment direction, initializing the gradient at the desired width from Equation 3.2, and decrementing the gradient as it is passed from block to block. This means that the interior blocks need not use information about the actual hydrostatic force; instead, they simply check if the gradient message they receive is greater than zero.

$$P = \rho g h \tag{3.1}$$

$$W_{des} = \alpha P + \beta \tag{3.2}$$

The algorithm is detailed in Algorithm 4. For exterior blocks, the algorithm is identical to the blind algorithm: for each direction, if the indicated position is valid, the water sensors give positive readings, and the direction is not the containment direction, the algorithm designates the site a valid attachment site.

The interior blocks make use of the hydrostatic width gradient that is propagated from the exterior blocks. The algorithm iterates through all 6 directions, and checks the validity of the position and that the hydrostatic width gradient is greater than zero. If both of these checks are positive, the algorithm designates the site in the given direction a valid deposition site.

Algorithm 4: Hydrostatic adaptive algorithm

```
Result: Valid attachment sites for a block
attachments ← [];
if block_type = exterior then
    foreach face_direction do
        neighbor ← GetNeighbor(face_direction);
        is_wet ← water sensors;
        is_valid ← CheckValidity(neighbor);
        if face_direction ≠ out & is_wet & is_valid then
            attachments ← [attachments,neighbor];
        end
    end
else
    foreach face_direction do
        neighbor ← GetNeighbor(face_direction);
        ok_width ← GetNeighborHydroWidth(out) > 0;
        is_valid ← CheckValidity(neighbor);
        if ok_width & is_valid & face_direction ≠ up then
            attachments ← [attachments,neighbor];
        end
    end
end
```

This algorithm results in a staircase or right-triangle shape, as seen in Figure 3.5.4. Hydrostatic pressure increases with water depth, so increasing water height causes an increase in not only the height of the structure, but also the width. This results in a structure that is continually stable regardless of environmental factors.

Table 3.2: Experiment 1: Adaptive Behavior with Hydrostatic Algorithm

Figure 3.6.X	channel width	water height	seed position	number of blocks	time steps
1	4	3	[14,2,1]	35	306
2	3	6	[16,1,1]	88	1,108
3	6	5	[14,2,1]	112	1,463
4	10	4	[10,8,1]	121	1,633
5	20	2	[12,10,1]	83	1,296
6	12	10	[14,6,1]	728	15,980

3.4 Experiments and Results

There are three key system capabilities that we aim to show in order to demonstrate adaptive construction in uncertain environments:

1. Adaptive behavior that generates effective structures in a variety of environments, using the same inputs
2. Automatic response to damage or to changing environmental factors
3. Scalability in construction time and intra-structure communications that supports almost limitless scaling of this system

3.4.1 Adaptive Behavior

In Experiment 1, we show that our system is capable of adapting to a variety of environments. We test this capability by choosing an algorithm from Section 3.3, and deploying our system using this algorithm in a variety of different environments. For this experiment, we use the hydrostatic algorithm - Algorithm 4 - and deploy the system in a channel-like setting, where a desired structure builds past the height of the water, and builds across the channel, touching each wall. Water behind the wall is drained 'downstream' in this scenario. We varied the width of the channel, the height of the water, and the position of the seed block, and documented the results in Table 3.2 and Figure 3.6.

3.4.2 Damage and Environment Response

To demonstrate the system's ability to quickly respond to unexpected damage to the structure, or to unexpected changes in the environment, we designed two experiments.

In Experiment 2, which is depicted in Figure 3.7, the system uses the hydrostatic algorithm (Algorithm 4) to build a floodwall in a channel setting with channel width 8 and water height 4. At timestep 1200 8 blocks (a 2x2x2 block cube) are instantaneously removed from the structure (Figure 3.7.2) by external

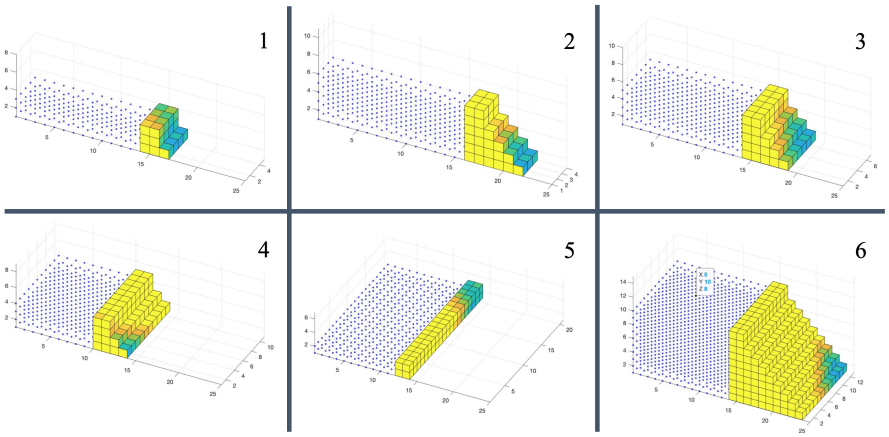

Figure 3.6: Resulting structures from Experiment 1, testing adaptation with the hydrostatic algorithm. The sub-figure numbers correspond to each of the test cases in Table 3.2.

forces - neither the smart blocks nor the actuator robot are directly informed of this damage; they must autonomously discover it. The system successfully patches the hole and continues to build the structure.

In Figure 3.7.1, the system has almost completed the structure. In 3.7.2, the 8 blocks mentioned above are removed without directly informing the system. In 3.7.3, after one timestep, the smart building blocks recognize that there is damage in the structure from their updated sensor readings. This can be seen by the color change of the blocks surrounding the hole, signifying the gradient of these blocks. The yellow color in 3.7.2 indicates the blocks have a high gradient, and are not near a valid deposition point. The dark blue color of the same blocks in 3.7.3 indicates that they are near a valid deposition point (the hole) and have begun propagating the gradient. In 3.7.3, water also begins reentering the previously drained area behind the wall through the hole. Water pools behind the wall in 3.7.4 and 3.7.5, as the actuator robot begins fulfilling the block requests, and patching the hole. In 3.7.6, the front layer of the hole is fixed, blocking the water again, and it again begins to drain from behind the wall. In 3.7.7 the structure is fixed, and in 3.7.8 the structure has been completed.

Experiment 3 addresses the ability of the system to react to changes in environmental conditions. In the floodwall application, this most naturally occurs in the form of changing water levels. In this experiment, we simulate floodwall construction in a channel-like setting with channel width 5 and initial water height 3 (Figure 3.8). Construction takes place normally, and shortly after the structure is complete (406 time steps), the water level is raised to 4. This causes the water to flow over the wall. The smart building blocks register this change and designate new deposition points, and the wall is grown to once again fulfill its function.

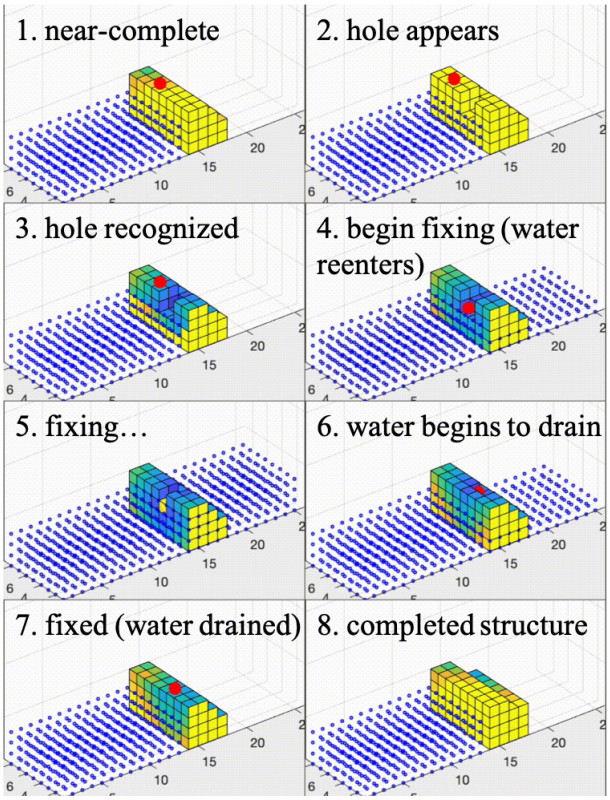

Figure 3.7: Experiment 2: Fixing a hole.

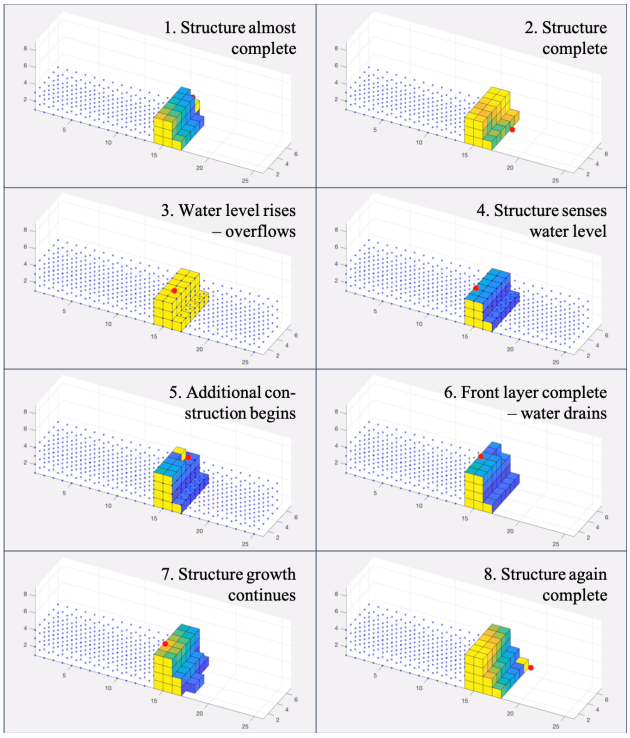

Figure 3.8: Experiment 3: changing water level.

3.4.3 Scalability

In Experiment 4, we analyze the scalability of each of the algorithms developed in Section 3.3. Scalability is a difficult concept to capture, especially in a single metric. For our purposes, scalability is defined as the ability of our system to function at any arbitrarily small or large scale. In this report, we focus not on the physical implementation of a system, which carries its own challenges at any scale. Instead, we aim to make the algorithmic aspect of our system scalable. Therefore, we consider scale in the number of building blocks, but not the size of building blocks; algorithmically, our system is block-size agnostic.

Our system is well-characterized in small scales, because this is where most testing and experimentation takes place. With the following experiment, we aim to characterize the system at arbitrarily large scales by investigating the relationships between the size of the structure being built, and a number of scalability metrics.

Metrics:

We use the following metrics to characterize the scalability of the system:

1. Wall Surface Area: water height $\times$ channel width, the wall surface area is a measure of structure size. It is an indicator of the minimum wall cross section that will hold back the water.
2. Number of Blocks: the number of smart blocks that are in the structure is the most representative metric for structure size.
3. Number of Time Steps: the measure of how long it takes the system to complete the structure. In one time step, the actuator can move one unit and the blocks can communicate to their immediate neighbors.
4. Average Number of Block to Actuator Messages per Time Step: The number of messages the actuator robot must handle per time step. In our paper, we consider a message to be a single integer.
5. Average Number of Block to Block Messages per Block per Time Step: The number of messages each block must handle per time step. Again, a message is a single integer.

These metrics for scalability come partially from [28], which uses time, number of block to actuator messages, and number of block to block messages as metrics.

Results:

To investigate the relationship between the metrics for size of the structure and the metrics for the cost of completion of the structure (time and number of messages), we ran the simulation repeatedly, varying the

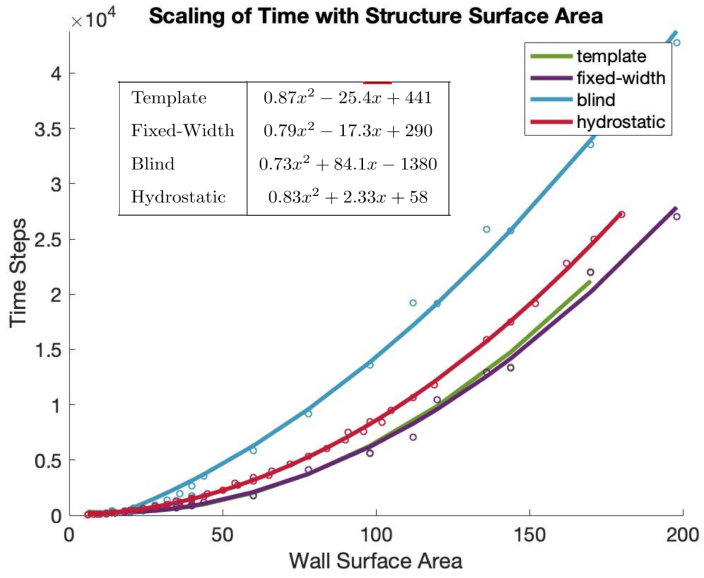

Figure 3.9: Plot of the number of time steps taken vs. the wall surface area of the structure, for each of the four algorithms.

width of the channel inside which the floodwall is built, and the height of the water. We varied from a width of 3 units and a water height of 2 units, to a width of 20 units and a water height of 9 units. In terms of the number of blocks in the final structure, this range encompassed more than two orders of magnitude. We found trend curves that encompassed these relationships and enabled us to extrapolate further. Figures 3.9, 3.10, 3.11, and 3.12 show the results of these experiments, and their trend line equations.

Figure 3.9 shows the relationship between the wall surface area of the final structure and the number of time steps taken. This relationship depicts how quickly each algorithm is able to complete the construction task, which is correlated to the shape of the final structure: algorithms which build smaller structures (e.g. fixed-width and template algorithms) will cover the same surface area in less time.

Figure 3.10 captures the relationship between structure size and time to build. Essentially, this characterizes the additional building time required by the specific structure shape that emerges from each algorithm.

Figure 3.11 shows the number of messages the actuator robot must handle as a function of size of the built structure (number of blocks). The y-axis metric is the number of messages the actuator robot received per time step. This data was difficult to fit, especially for the hydrostatic algorithm (see Table 3.3 for fit performance). The trend lines are hyperbolic.

Figure 3.12 demonstrates the scaling of each block's message load with the size of the final structure. The y-axis metric is the number of messages each block receives per time step, calculated by dividing the total number of block-to-block messages by the total number of blocks, and again by the number of time

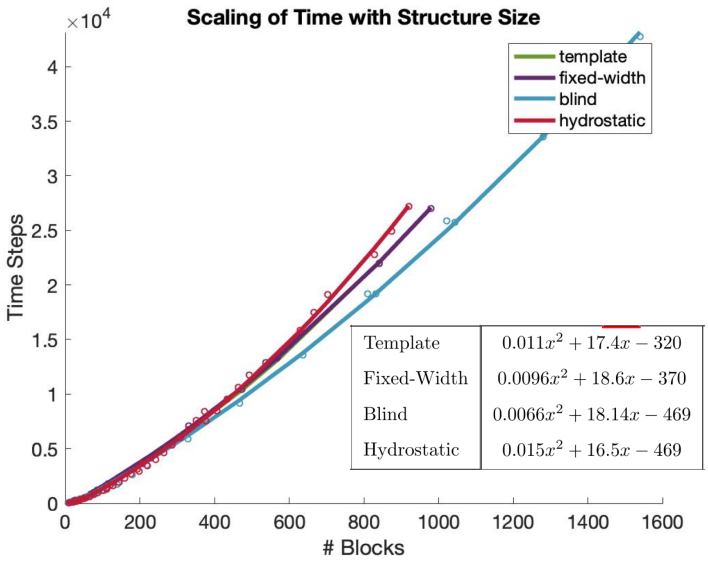

Figure 3.10: Plot of the number of time steps taken vs. the number of blocks in the final structure for each of the four algorithms.

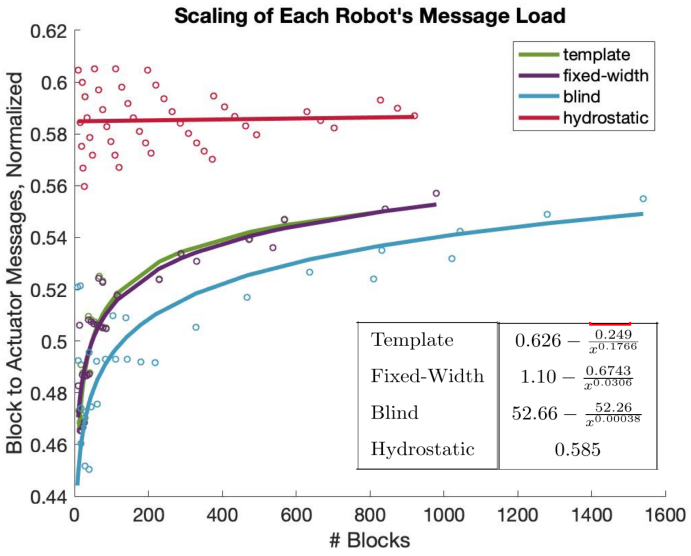

Figure 3.11: Plot of the number of messages sent between each actuator robot and smart building blocks per time step vs. the number of blocks in the final structure.

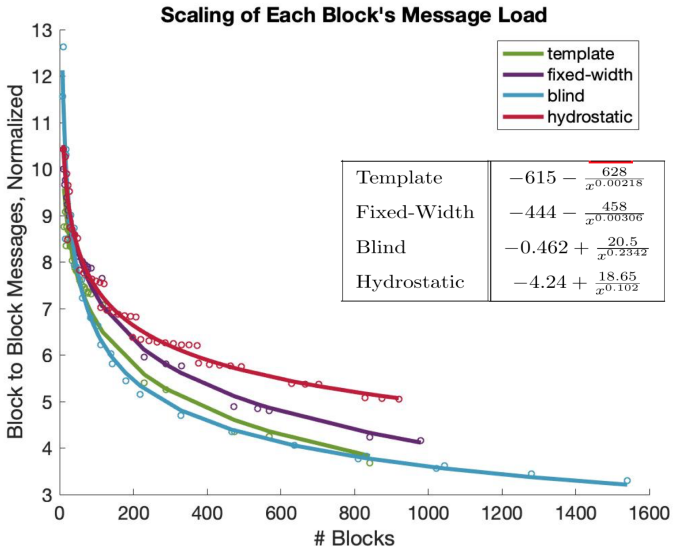

Figure 3.12: Plot of the number of messages each smart block handles per time step vs. the number of blocks in the final structure.

	Figure 3.9	Figure 3.10	Figure 3.11	Figure 3.12
Template	0.9941	0.9995	0.8613	0.9589
Fixed-Width	0.9943	0.9993	0.8196	0.9851
Blind	0.9948	0.9993	0.3019	0.9698
Hydrostatic	0.9992	0.9983	0.0017	0.9739

Table 3.3: r^2 values for the trend equations in Figures 3.9, 3.10, 3.11, and 3.12. Poor values are highlighted in red.

steps. The trend lines for this data are hyperbolic.

3.5 Discussion

3.5.1 Adaptive Behavior

Experiment 1 shows successful adaptive behavior. We use the hydrostatic algorithm to build structures in a variety of environments. We give the system the exact same input for every trial: the direction in which to contain the water. We vary the channel width, water height, and the position in which the seed block is placed. Changing these variables, we are not able to find a scenario in which the hydrostatic adaptive algorithm does not complete a successful structure.

Note that the overall shape of the system varies according to the inputs: a higher water level (Figure 3.6.2,3.6.6) will result in a wider structure, as the blocks react to larger hydrostatic forces. A lower water level (Figure 3.6.5) will result in a narrow cross section. This demonstrates the efficiency of the algorithm

at creating a robust structure: it does not build more than is necessary, because it is responding directly to the environmental parameters captured by the sensors.

Additionally, the system is agnostic to where the seed block is placed. The exact coordinates of seed block placement are shown in Table 3.2, but the qualitative effects can be seen in the gradients in Figure 3.6. The yellow, which represents high gradient values, is nearer the seed location, because these areas are built first. The darker colors are the last blocks that are placed. In the first, second third, and sixth image, the robot builds the rear right portion of the structure last. In the fifth image, the right side of the structure is the last to be built, and in the fourth image, the back left portion of the structure is the last to be built. The final state of the system is not impacted by the order of building that results from different seed block placements.

As previously mentioned, the current state of the art in the literature, as presented in [22], [28] is represented by the template algorithm. The template algorithm will fail to adapt to different environmental conditions: the template has to be designed for each environment, and will not be effective in an environment for which it is not designed. If the water level in an environment is higher than in the environment for which the template is designed, the wall will overflow. If the water level is much lower, the template will be far too large, resulting in unnecessary expense.

Compared to the fixed-width algorithm, the hydrostatic algorithm results in a more adaptive system. Again, the width must be known for each specific environment. If water levels are smaller than expected, the fixed-width algorithm will be very short and fat. If they are higher than expected, it will be dangerously thin (Figure 3.13).

3.5.2 Damage and Environment Response

In Experiments 2 and 3, we analyzed how the system responded to damage that was inflicted on the structure, and to changes in the water height during construction.

Damage Repair:

Figure 3.7 shows Experiment 2, where a hole is created instantaneously in the structure, and the system must recognize the hole and repair it. This capability is possible in other systems, including those that do not use smart building blocks. The key advantage of our system is the speed of repair. Because the building blocks are able to recognize the damage after one timestep, and propagate this information at the speed of block-to-block communication, the damage can be repaired orders of magnitude faster. This speed increase for construction, though not specifically for repairing, was shown in [23].

In our experiment, the damage was recognized in one timestep, and the gradient was propagated to the robot position in 12 timesteps.

Note that the shape of the completed structure in Figure 3.7 is not the typical shape we expect from the hydrostatic algorithm: it is wider than the staircase shape normally seen. If an internal block in a given y-axis (front to back) row is placed before the external block in this row, the block may request additional deposition points that violate the hydrostatic width constraint. These extra blocks will cause an additional x-axis row (across the channel) to be built. This behavior results in over-compensation for damage, a side-affect that could be eliminated with a simple mechanism to stop irrelevant deposition requests. However, it is advantageous to over-repair damaged areas: it is reinforcement against further damage. Therefore, we do not remove the behavior.

All three of the other algorithms are capable of fixing damage of this type in the same manner and with the same efficiency. Under each of the four algorithms, the system is robust to many different types of damage. The system will fix a simple hole, as shown in Experiment 2. The system can also handle holes that are only on the front or back surface of the structure, and which don't penetrate the entire structure. The system can handle large scale disturbances, like most of the building blocks being washed away, perhaps separating a group of already placed building blocks from the seed and depot entirely. The system will continue building and complete the structure.

Environment Response:

In Experiment 3, we build a structure using the hydrostatic algorithm, and raise the water level after construction is completed. This causes the floodwall to overflow, spilling water back behind the wall. This spurs more structure growth, with construction occurring first atop the wall, and then back behind to further reinforce. This results in a final structure that again satisfies the function of water containment.

The hydrostatic algorithm guarantees stability of the floodwall structure even as environmental factors change by making use of hydrostatic force measurements to determine the width of the wall.

The blind algorithm (Algorithm 2) is equally as adaptive to this type of environmental change, but its structure may not be as robust: because it is not responding directly to the forces on the structure - it only builds until water drains from behind it - it may not make a structurally sound shape. For example, in Figure 3.5, the width at the far end of the structure is only two building blocks. This is relatively small for a water height of eight units.

The template algorithm is not adaptive to this type of environment change at all. If the water height increases and it overflows, it will not detect anything wrong, and the system will not change. The same occurs if the channel were to widen: water would flow by the sides of the structure, but the template algorithm

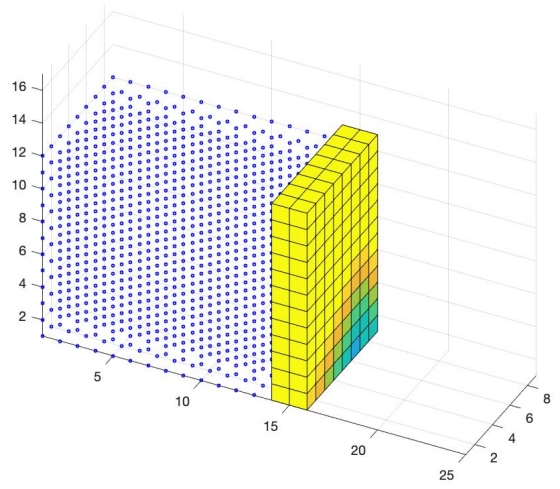

Figure 3.13: The structure resulting from the fixed-width algorithm with a specified width of two, for low floodwater circumstances. The water height was then raised to 12 to simulate unexpectedly high flooding.

would not register this change.

The fixed-width algorithm is adaptive to this type of environmental change, but like the blind algorithm, it does not guarantee a stable structure. For example, if the water height is expected to be small, and a width of two units is chosen, the structure will remain two units wide regardless of other environmental changes. If the water height were to increase drastically, the resulting structure (shown in Figure 3.13) would be absurdly tall and skinny, and clearly not stable.

3.5.3 Scalability

Scalability is an important and difficult to characterize metric for a system such as this. In modular robotics, greater benefits are realized as scale of the individual module decreases, and the number of modules in the system increases. This is the same for our system: increase in the number of building blocks results in structures with higher fidelity, smaller cracks, and more redundancy.

Our first measure of scalability is how the structure build time scales with the surface area of water that the structure must contain. As shown in Figure 3.9, all four algorithms scale quadratically with surface area.

The relationship between these two metrics reflects the shape that emerges from each algorithm. The blind algorithm scales the worst, because it makes the largest structure for a given water height and width. The results from the template and fixed-width algorithms are similar; in this experiment, they were both implemented with $width = max(floor(\frac{waterheight}{2}), 2)$, and so very similar structures resulted from each. Their scaling with surface area will depend largely on the widths chosen. The hydrostatic algorithm falls

between: it builds a larger and more robust structure than the template and fixed-width algorithms for most cases, but a smaller structure than the blind algorithm.

Quadratic performance (with the square term less than 1) is reasonable performance for construction of structures that are roughly cubic in shape. If faster performance is necessary, a greater number of actuator robots could be used.

In Figure 3.10, we directly compare the structure size with the time to completion of the structure. In our system architecture, where an actuator robot must travel the distance from the depot to the deposition point and return every time a block is placed, we expect this relationship to be slightly worse than linear, due to the travel time of the robot.

This expectation is accurate. Each of the four algorithms scales quadratically in these two metrics with very small x^2 coefficients, between 0.0066 and 0.015. This scaling can be further improved, again, by using more actuator robots.

The difference between each of the four algorithms in this relationship is small, and a function of the structure shape. The blind algorithm scales best in these metrics, as the robot is always building as close to the depot as possible. The hydrostatic algorithm scales worst, as its shape is distributed such that the robot must travel farther throughout construction.

Near-linear scaling is expected and acceptable performance in this metric. In a system where a single agent builds a structure, better than linear scaling would indicate that each block is placed *faster* as structure size increases. But our actuator robot can only carry one block, and as the structure size grows, the distance the robot must travel increases: linear scaling is the best case scenario for a single-robot implementation.

In Figure 3.11, we observe the scaling of the number of messages that each actuator robot must handle per timestep with the number of building blocks. This data is difficult to fit meaningfully; in Table 3.3, the poor r^2 values are highlighted in red. However, hyperbolic curves were able to best characterize the data, with the exception of the hydrostatic algorithm, which was best fit as a constant. These trend equations, which were generated by a numerical fitting algorithm in the Matlab curvefitting toolbox, vary widely, but we note that large differences in coefficients only result in significant changes in the curve as the number of blocks increase to the order of 10^7.

The important trend to note in this data is that for every algorithm, the actuator robot message load is near constant as number of building blocks increases. Even the worst-scaling algorithm, the fixed-width algorithm, scales extremely well: the number of messages that each robot must handle per time step does not exceed 1 until the number of blocks is $\approx 10^{27}$. Since each robot should be capable of handling many more than one message per time step, this is a highly favorable result.

In Figure 3.12, we look at how the number of messages that each block receives per time step scales with

the number of blocks in the structure. Similar to the trends in 3.11, the data is best characterized with hyperbolic curves. These decreasing hyperbolic curves fit the data with $r^2 > 0.97$ (Table 3.3), and indicate very favorable scaling of block message load. The *decreasing* curves indicate that as the number of blocks in the system increases, each block must handle *fewer* messages per time step. As above, there is large variation in the coefficients of the trend lines, but this does not significantly affect the curves in the range of block number that we analyzed.

Based on the results from these scaling analyses, we conclude that the system is highly scalable: as size of the structure increases, the time of construction increases slightly worse than linearly, and the message load that the actuator robots and the smart building blocks must handle per time step decreases. Therefore, the scale of the system is limited by the worse-than-linear scaling with time. This is expected with a system in which the construction materials must be transported to deposition points and installed. This limiting factor can be mitigated by using multiple construction robots. This merits further investigation, but we expect that as the number of actuator robots approaches the number of blocks in the structure, better-than-linear performance could be achieved in time.

3.6 Conclusion

We presented a system architecture that successfully builds a floodwall with smart building blocks, and four sets of local rules that each result in successful emergent structures.

In Experiment 1, we showed that our system can adapt to varied environments, a novel achievement for smart building block based systems. In Experiment 2, we demonstrated that our system can rapidly repair itself when damaged. In Experiment 3, we showed another novel capability for a smart building block system: response to changing environmental factors. Finally, in Experiment 4, we showed that the system is scalable in terms of the message load for robots and building blocks, and near-linear in time.

3.7 Future Work

The immediate extensions of this work are as follows:

1. More thorough comparison with modular robot systems and passive building block systems, in terms of construction speed and capability.

2. Provability of construction: [28],[5],[7] prove that their system will always be capable of building a specific class of structures. We would like to prove this for our system. The difficulty of this proof

is that we use sets of local rules, and so do not have a clear definition of our structure's final shape. Further, we would like to prove that the structure is capable of repairing any arbitrary damage, in terms of blocks removed.

3. Implementing sets of local rules that make use of more direct measures of structure stability, such as strain sensor data.

Longer-term, interesting investigations include:

1. Investigation and quantification of robustness of the system to sensor failure and robot failure, including robustness to noise in sensor data, communications, and Byzantine fault tolerance.

2. Implementation of multiple actuator robots working together to build the system, and an investigation of how this impacts the scalability of the system in time and communications.

3. Investigation of a hybrid system of smart building blocks and passive, inert blocks: how many building blocks do we need to fully embody the system, and achieve adequately responsive behavior?

The long-term goal of this research is to enable more complex construction in unstructured environments. To do this, we envision a system that makes use of a variety of local rule sets as construction pieces, and makes templates composed of these construction pieces. For example, there might be a local rule set for a wall, for a pillar, and for a roof. These could be combined to make a roofed structure (Figure 3.14).

Combining sets of local rules in this way would allow adaptive construction, self-repairing, and scalability of each individual section, enabling complex construction in unstructured environments. It would also allow simple specification of more complex structures.

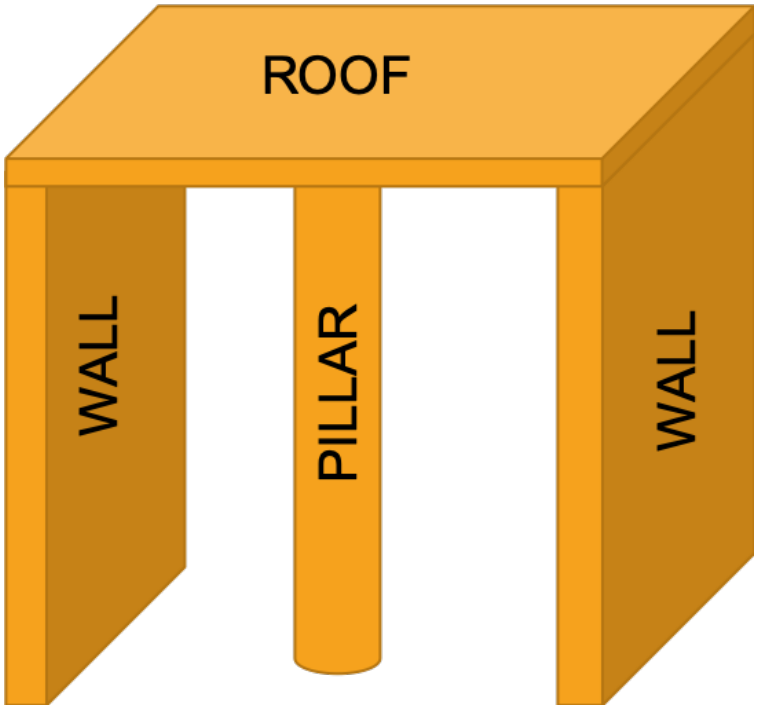

Figure 3.14: An imagined template of local rule sets, which would result in the construction of a roofed structure in an adaptive manner.

Bibliography

[1] "Occupational safety and health administration commonly used statistics." https://www.osha.gov/oshstats/commonstats.html, 2018. Accessed 2019-05-19.

[2] V. M. Pawar, R. Stuart-Smith, and P. Scully, "Toward autonomous architecture: The convergence of digital design, robotics, and the built environment," *Science Robotics*, vol. 2, no. 5, p. eaan3686, 2017.

[3] C. Trapasso, "How 3-d printers will lower prices, make fantasies real, and transform the housing market." https://www.realtor.com/news/trends/3-d-printed-homes/, Aug 2016. Accessed 2019-05-19.

[4] "Vice video: This bricklaying robot can build walls faster than humans'." https://video.vice.com/en_us/video/this-bricklaying-robot-can-build-walls-faster-than-humans/5967c4584d7c0670791926a4. Accessed 2019-05-19.

[5] N. Napp and R. Nagpal, "Distributed amorphous ramp construction in unstructured environments," *Springer Tracts in Advanced Robotics*, vol. 104, pp. 105–119, 2014.

[6] M. S. D. Silva, V. Thangavelu, W. Gosrich, and N. Napp, "Autonomous Adaptive Modification of Unstructured Environments," *Robotics: Science and Systems*, 2018.

[7] M. Saboia, V. Thangavelu, and N. Napp, "Autonomous Multi-Material Construction with a Heterogeneous Robot Team Autonomous Multi-Material Construction with a Heterogeneous Robot Team," no. September, 2018.

[8] C. A. Parker and H. Zhang, "Collective robotic site preparation," *Adaptive Behavior*, vol. 14, no. 1, pp. 5–19, 2006.

[9] T. Tosun, J. Daudelin, G. Jing, H. Kress-Gazit, M. Campbell, and M. Yim, "Perception-Informed Autonomous Environment Augmentation With Modular Robots," 2017.

[10] K. C. Galloway, R. Jois, and M. Yim, "Factory floor: A robotically reconfigurable construction platform," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2467–2472, 2010.

[11] D. Hjelle and H. Lipson, "A robotically reconfigurable truss," *2009 ASME/IFToMM International Conference on Reconfigurable Mechanisms and Robots*, 2009.

[12] S. K. Yun and D. Rus, "Adaptive coordinating construction of truss structures using distributed equal-mass partitioning," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 188–202, 2014.

[13] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction with quadrotor teams," *Autonomous Robots*, vol. 33, no. 3, pp. 323–336, 2012.

[14] T. Soleymani, V. Trianni, M. Bonani, and F. Mondada, "An Autonomous Construction System with Deformable Pockets Technical Report No .," no. January, pp. 5–9, 2014.

[15] D. J. Christensen, "Experiments on fault-tolerant self-reconfiguration and emergent self-repair," *Proceedings of the 2007 IEEE Symposium on Artificial Life, CI-ALife 2007*, pp. 355–361, 2007.

[16] M. H. Hansell, *Animal architecture*. Oxford University Press, 2008.

[17] K. Petersen, R. Nagpal, and J. Werfel, "TERMES: An Autonomous Robotic System for Three-Dimensional Collective Construction," in *Robotics: Science and Systems VII*, 2011.

[18] N. Napp, O. R. Rappoli, J. M. Wu, and R. Nagpal, "Materials and mechanisms for amorphous robotic construction," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4879–4885, 2012.

[19] J. Werfel, D. Ingber, and R. Nagpal, "Collective construction of environmentally-adaptive structures," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2345–2352, 2007.

[20] J. Werfel, K. Peterson, and R. Nagpal, "Designing Collective Behavior in a Termite-Inspired Robot Construction Team," *Science*, vol. 343, no. February, pp. 754–758, 2014.

[21] Y. Terada and S. Murata, "Automatic modular assembly system and its distributed control," *International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 445–462, 2008.

[22] J. Werfel and R. Nagpal, "Extended Stigmergy in Collective Construction," *IEEE International Conference on Intelligent Robots and Systems*, 2006.

[23] J. Werfel, Y. Bar-Yam, D. Rus, and R. Nagpal, "Distributed construction by mobile robots with enhanced building blocks," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, no. February, pp. 2787–2794, 2006.

[24] H. Bojinov, A. Casal, and T. Hogg, "Emergent structures in modular self-reconfigurable robots," *Proceedings-IEEE International Conference on Robotics and Automation*, 2000.

[25] K. D. Kotay and D. Rus, "Generic distributed assembly and repair algorithms for self-reconfiguring robots," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2004.

[26] M. Allwright, N. Bhalla, C. Pinciroli, M. Dorigo, M. Allwright, N. Bhalla, C. Pinciroli, and M. Dorigo, "Towards Autonomous Construction using Stigmergic Blocks," 2017.

[27] J. Kubica, A. Casal, and T. Hogg, "Complex Behaviors From Local Rules in Modular Self Reconfigurable Robots," 2001.

[28] J. Werfel and R. Nagpal, "Three-dimensional construction with mobile robots and modular blocks," *International Journal of Robotics Research*, vol. 27, no. 3-4, pp. 463–479, 2008.

[29] "Rapid deployment flood wall." `http://www.geocellsystems.com/`. Accessed 2019-05-19.