Robotic Construction Using Intelligent Scaffolding

by

Albert Enyedy

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Masters of Science

in

Robotics Engineering

by

May 2020

APPROVED:

Dr. Carlo Pinciroli, Advisor

Dr. Zhi Li, Assistant Professor

Dr. Gregory Lewin, Assistant Professor

Abstract

Construction is a complex activity that requires the cooperation of multiple workers. Every year, construction activities cause injuries and casualties. To make construction safer, new solutions could be provided by robotics. Robots could be employed not only to replace human workers, but also to make construction in harsh environments safe and cost-effective, paving the way for enhanced underwater infrastructure, deeper underground mining, and planetary colonization.

In this thesis, we focus on the topic of collective construction, which involves the cooperation of multiple robots, by presenting a collective robot construction method of our own. Collective construction can be a more viable option than employing individual, complex robots, by potentially allowing the effective realization of large structures, while offering resilience through redundancy, analogous to insect colonies.

Our approach offers a novel solution in the design trade-off between choosing the number of robots involved vs. the complexity of the robots involved. On the one hand, capable and complex robots are expensive, limiting the cost effectiveness of realizing large swarms which provide redundancy and increase the system's resilience to faults. On the other hand, simple and inexpensive robots can be manufactured in large numbers and offer high redundancy, at the cost of limited individual capabilities and lower performance. We use two types of robots: intelligent scaffolding and worker robots. The intelligent scaffolding acts as regular scaffolding, allowing the worker robots to navigate the structure they assemble, while also guiding and monitoring the construction of the structure. The worker robots move and connect scaffolding and building material while only knowing the local commands necessary to complete their task. This approach is loosely inspired by termite mounds, in

which termites use the process of stigmergy in which they mark construction pellets with pheromones to affect the progress of construction, while navigating the structure that they build. Thanks to intelligent scaffolding, construction robots have a simple design that allows minimalist onboard computation and communication equipment.

In this thesis, we produced a minimum viable prototype demonstrating this concept. Intelligent scaffolding is realized through smart blocks that can be laid and connected to each other. The smart blocks are capable of simple computation and communication once laid. The construction robot uses local navigation methods by line-following across the scaffolding and building blocks of the system. The blocks and construction robot both have a modular design, simplifying the process of manufacturing and repairs while maintaining a low cost. The robot and blocks use magnets to increase the margin of error during block manipulation and allow for the assembly and removal of scaffolding as well as its reuse between build sites. To communicate with the robot, the intelligent scaffolding blocks send local IR signals, similar to TV remote signals, when the robot is on top of them, minimizing the risk of global interference and keeping the system portable. To monitor the connectivity of the system throughout the life cycle of the structure, electrical connections run through each of the blocks, which indicate the status of the structure and can be used to diagnose the location of breaks in the structure for maintenance.

Acknowledgements

I would like to first thank my thesis advisor Professor Carlo Pinciroli of Worcester Polytechnic Institute (WPI) for his support and advice throughout the course of this thesis and my education at WPI. He always made time to provide me with feedback even during his busiest weeks, and provided me with valuable insight on graduate studies as a whole. He also taught me how to become a better writer and presenter, skills which I have been using ever since.

I would also like to thank my thesis review committee member Professor Greg Lewin of WPI for taking the time to review my thesis. He also took time to meet with me to provide feedback on my robot design, allowing me to make necessary improvements that would allow for the success of the navigation methods.

I would like to thank my thesis review committee member Professor Zhi Li of WPI for taking the time to review my thesis as well. She also provided me with valuable advice and insight on graduate studies, which I have taken to heart in my pursuit of further education.

I would like to thank Felix Sanchez of WPI for co-authoring the 2018-2019 Major Qualifying Project that developed the prototype of the intelligent scaffolding blocks and robot that I used for the basis of this thesis. He designed the intelligent scaffolding blocks and developed the I2C method that I used in this thesis, and I'm glad he could see the successful full demo implemented in this thesis in action.

Finally, I would like to thank my parents and my brother for their unwavering support, and my partner, Jillian Eckard, for always being there for me throughout the years. They made accomplishing this goal of mine possible.

Author

Albert Enyedy

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We rely on construction to build and maintain our infrastructure. Multiple workers cooperate to create complex structures in potentially dangerous environments, such as working open-air on the girders of high-rise buildings or creating a tunnel. Construction results in many injuries, both fatal and nonfatal, every year. In 2018 alone, according to the Bureau of Labor Statistics, almost 200,000 workers were injured in the construction industry [3], with about 1000 deaths [4]. However, construction must continue, despite injuries and deaths, since the completion of the project sites is required to provide important services such as housing, new transportation routes, or repairs of old ones. Thus, new construction solutions are sought out to improve the effectiveness but also the safety of construction sites. One such solution, and the solution we focus on in this thesis, is the use of robots to perform construction tasks.

To increase the safety of construction sites, new robotic construction solutions could be implemented. Robots enable the option of removing human workers from the more dangerous sections of the construction site while also providing the opportunity to realize new construction methods that humans do not or cannot perform.

Robots can also cost-effectively aid construction in harsh environments, providing technology to enhance underwater and underground infrastructure as well as to pave the way for future advancements, such as planetary colonization. For example, to build structures on the Moon or Mars, a method of 3D printing using novel concrete for additive manufacturing is developed by Khoshnevis *et al.* [15], in which a KUKA robot arm is used to extrude structures. A similar robot arm could be sent to the Moon or Mars to build structures for astronauts.

Construction robots can take multiple forms, such as large, complex, single robots or smaller, simpler groups of robots that cooperate with each other. Such groups of smaller, simpler robots are classified as robot swarms in the field of swarm robotics, as described by Brambilla *et al.* in [5]. Robot swarms use simple behavioral rules and local communication and interactions to leverage their collective capabilities. Other main tenets of swarm robotics are the use of decentralized control, in which each robot makes its decisions based on local data as opposed to instructions from a central source; and robots have no access to global data, instead relying on their local perception of the environment. For example, to use multiple robots to push an object, as shown in Campo *et al.* [7], robots move in a common direction by communicating their individual direction to their neighbors, until they find the object to push. Then the robots use simple rules dictating a pulling behavior to begin moving the block to the goal.

We will focus on swarm robotics applied to construction problems to implement a "collective robot construction" (CRC) system, which uses various swarm principles to build structures. We look to the current state of the art in CRC systems, and find that each approach to collective robot construction has its own drawbacks that limit their potential effectiveness in real construction sites outside of the lab.

The quadrotor construction systems implemented by Lindsey *et al.* [17] or Augugliaro

et al. in [2] use multiple quadrotors with high-level position control, calculated by an external computer reading motion-capture data from a system of cameras that sends commands to coordinate the robots. An implementation using global vision-based sensing limits the robots' effective range by requiring them to stay within range of the motion-capture system's cameras and the command computer's signal range, thus reducing potential effectiveness in a real construction site. The work by Parker *et al.* in [19] uses robots to create a nest by bulldozing away material, thus effectively removing material from a build site. While being one of the few CRC robot systems that removes environmental material at a construction site, this CRC system cannot perform construction using a building material of its own which limits its effectiveness for the process of building structures (as a skyscraper is not built by pushing dirt). And once a structure is built, no CRC system has implemented a method for robots to remain present throughout the life cycle of the structure to perform tasks such as maintenance or demolition.

Thus, to implement effective CRC systems that avoid dependence on limiting global sensing and communication methods while maintaining a simple design, we look to nature for inspiration. Termites, such as *Macrotermes subhyalinus (Rambur)* researched by Bruinsma in [6], build massive structures relative to their body sizes using only local communication methods, for example by emitting pheromones that other termites sense when nearby. To build the termite nest, the queen emits a build pheromone that notifies the workers to begin building, and specifies the distance to and orientation of their construction site [6]. The specific paths taken to the construction site are determined by the trail and cement pheromones laid by the worker termites traveling to and from the build site and placing their building material in the structure. This is an example of *stigmergy*, a concept introduced by Grassé's research on termite construction [12], in which markers left on the built

portion of the structure (or simply the structure itself) serve as indicators that guide the placement of the rest of the construction material in the structure. Thus, the termites effectively communicate with each other the paths required to reach the construction sites and whether more construction material must be placed on the structure. The queen also notifies the workers of when to build, while only communicating locally with the nearby termite workers that can sense the build pheromones. CRC systems such as Werfel *et al.*'s TERMES [26] and Allwright *et al.*'s SRoCS [1] were directly inspired by these construction methods of termites.

In this thesis, we draw direct inspiration from the termite-inspired TERMES and SRoCS systems by seeking to combine aspects of both systems. In TERMES, a structure is designed in software first, and then the instructions to build the structure are created by an offline compiler which then is provided to the worker robots. The worker robots thus know the plan of the final structure, and build it block by block using local sensing and navigation methods. The structure is navigable, like a termite mound, thus guaranteeing successful completion of the structure as the robots travel across the blocks they have just placed [26]. The SRoCS system uses stigmergic blocks to communicate with the worker robot and other blocks, providing environmental information and building instructions which are important as the SRoCS system drives across the environment and does not navigate across the structure it builds. Thus, the worker robots do not know any final structure designs and follow commands from the stigmergic blocks instead. The blocks can indicate to the robots that a specific structure based on a pattern should be built at the specified location around them, thus offloading the structure's plan to the blocks as opposed to storing them on the robot like in TERMES. The robot uses local sensing methods to detect obstacles and features in the environment, also aiding in guiding the construction process [1].

1.1 Problem Statement

By combining methods from the TERMES and SRoCS systems, we seek to implement a system inspired by termites that makes full use of stigmergy by offloading the structure's plan onto a system of intelligent scaffolding blocks (which serve as the stigmergic blocks in SRoCS) that the robot also travels on (like the building material in TERMES), thus reaping the benefits of both stigmergic construction material and navigable structures. Thus, our robot must use local sensing methods for communication with the scaffolding as well as for navigation. Our solution must also offload as much computation as possible from the worker robot to the intelligent scaffolding units, to implement stigmergy by having the blocks guide the construction process. The intelligent scaffolding blocks and worker robots should have a modular, minimalist design as well, to simplify the process of assembling them in greater numbers at a low cost.

The robots should be able to manipulate the intelligent scaffolding units such that when the structure is complete, the scaffolding units can be removed and brought to the next build site, with the remaining structure left behind. However, the option to keep the scaffolding in the structure should be left open, such that the scaffolding can detect changes in the structure during its life cycle and issue repair commands to the robots to maintain the structure's integrity. Thus, the scaffolding must also have a method for sensing the state of the structure and whether any unexpected changes have occurred.

1.2 Contributions

We present a minimum viable prototype of a construction system that is the first system to leverage both stigmergy and navigable structures. Our system opens the

way for new research into construction algorithms that, using only the TERMES [26] or SRoCS [1] systems, were not previously possible.

Based on the requirements listed in the problem statement, our minimum viable prototype showcases a physical implementation of the proposed collective robot construction solution. The minimum viable prototype uses a system of a single manipulator *worker robot* with a set of two *intelligent scaffolding blocks* (one *seed* home position block, and one *leaf* block to place in the structure) and three intermediate *build material blocks*. The construction units are referred to as blocks in reference to their block-shaped design and how they tessellate in the same manner as blocks would.

Reminiscent of the stigmergic construction method used by termites, we offload the knowledge of the construction plan to the intelligent scaffolding blocks as opposed to keeping it on the robot; the intelligent scaffolding blocks distribute construction commands that the worker then follows. The intelligent scaffolding blocks and build material blocks also provide the robot with a navigable structure, simplifying the complex construction site environment into a discrete grid of reachable locations. Thus, we achieve our goal of simplifying the computation requirements of the robot, enabling us to effectively implement a minimalist robot design that allows the system to be easily expanded to make use of multiple robots while allowing for cost-effective and straightforward replacement of damaged robots.

The minimum viable prototype uses only local communication methods to transfer commands between the intelligent scaffolding blocks and the worker robot. The robot also only uses local sensing for navigation, orienting itself by line following across the surfaces of the structure. By avoiding global sensing and communication methods, our CRC system can work in a wide variety of environments with minimal risk of environmental interference while adhering to the principles of swarm robotics.

The worker robot also manipulates its own scaffolding blocks, thus completing the scaffolding removal capability required in our problem statement. This ability allows the scaffolding system to be picked up and reused at new build sites without the need for developing more scaffolding units, providing economic and environmental incentives for using the concepts introduced by our CRC system.

The intelligent scaffolding blocks and build material blocks also have electrical connections on each of their sides, allowing for the seed block to power its leaf block and for the blocks to communicate with each other. Once a leaf block is placed in the structure, it sends a signal to the seed block indicating that it is now part of the structure. Since a connection between the two has now been made, the seed block can sense if parts of the structure become disconnected by recognizing that the leaf block's signals are no longer being received. As such, the opportunity for keeping our robotic construction system in the life cycle of maintaining and eventually demolishing a structure is presented.

1.3 Outline

In this thesis, related works which influenced our design choices are described in Chapter 2. In Chapter 3, we describe the previous work that led to the current implementation of our swarm construction using intelligent scaffolding method, as well as the improvements made during the course of this thesis that enabled the completion of the minimum viable prototype. In Chapter 4 we describe the experiments that were done to test the effectiveness of the minimum viable prototype. The limitations of this method, as well as future works based on it, are discussed in Chapter 5.

Chapter 2

Related Works

Collective robotic construction methods can be realized in a variety of ways, many of which are inspired by nature. One of the most popular CRC algorithms is stigmergic construction [20], in which a set of markers, whether stigmergic blocks [1], intelligent scaffolding blocks (this thesis), or the status of the structure [18], guide the rest of the construction process. These indicators convey information to the robots about where to place more build material or whether the structure is complete. Stigmergic construction also allows for the realization of completely decentralized control methods by providing robots with a guide for the construction plan while using only local communication or sensing.

Another method for organizing CRC methods is using global sensing and control. These methods often use global vision-based sensing such as motion capture to gather data on the entire construction site, which is then used to organize the worker robots. Quadrotor construction methods ([17], [2]) often use global sensing and control, to locate the robots and ensure that they do not crash into each other while also keeping track of the goal locations for assembling the structure.

When designing CRC methods, it is important to consider the type of material

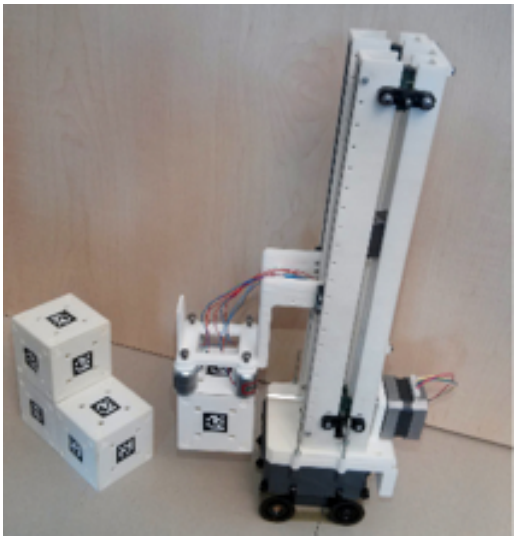
being used. When using discrete material, such as bricks or custom building blocks, the robot can be codeveloped with the target material in mind, and can be simpler due to the simple geometric nature of the material [20]. Systems such as TERMES [26] and SRoCS [1] use discrete building material. Continuous material, such as concrete, extruded material, or nylon thread [24], can also be used to build structures but require different methods for handling by robots. Sometimes, even the robots in the system themselves are the building material [13].

Many CRC systems, whether using global sensing and control or local sensing and control, make use of a "seed", which is a location in the construction site that each of the worker robots can use as a reference for locating themselves with respect to the build site as well as locating other points of interest in the build site. Some construction methods (i.e. stigmergy) use the seed to convey construction information, such as commands for building a structure based on a template ([1], [13]).

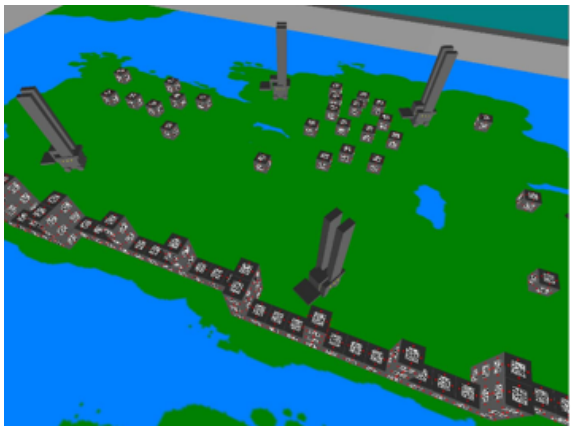
2.1 Stigmergic Construction Methods

The SRoCS system [1] uses multiple mobile manipulator robots (BeBots [14]) and stigmergic building blocks, as shown in Fig. 2.1a to assemble structures using decentralized control by having the blocks act as indicators to guide construction.

The robot has an elevator-like lift with a gripper to enable the stacking of the cube-shaped stigmergic blocks, and the blocks have magnets on each corner to enable self-alignment with other neighboring blocks. The blocks communicate with the robots through a system of RGB LEDs and NFC (Near Field Communication), with 2D barcodes for the robots' cameras to identify on each side. The robots thus can see if a block is a seed block requesting more blocks for a pattern design based



(a) SRoCS stigmergic blocks prototype and BeBot



(b) SRoCS simulation of constructing a wall around a river

Figure 2.1: SRoCS stigmergic construction system

on the color of the RGB LED. Once the robot places a block, it uses the NFC system to communicate to the block that it should change color to indicate that it is a build material block, thus indicating to the other robots that that unit of material has already been placed in the pattern, and they should place their blocks in another location to complete the structure instead. This system of stigmergy allows the SRoCS system to build structures that can conform to environmental features, as shown in simulation in Fig. 2.1b, by using previously placed blocks and local sensing to figure out block placement. To localize themselves in the build site, the robots detect the barcodes on the blocks and can thus find where they are with respect to the structure.

The work of Napp *et al.* [18] uses stigmergic construction methods to build a ramp. The robots have a target ramp angle they seek to produce, which they must reach by using foam, which can fill gaps and crevices in structures then harden to be navigable by robots. The stigmergy in the structure is the height of the ramp—if the robot senses that the height of the ramp is not high enough after placing its material

or another robot places material, then the robot knows that it must add material to the structure to make progress towards reaching the goal. The structure produced by the robots is also navigable, guaranteeing that the robots can traverse the structure they construct to reach every location required for placing build material.

2.2 Global Sensing & Control Systems

The work of Lindsey *et al.* [17] constructs rectangular structures using modular parts called nodes (corners of cubes) and members (edges of cubes), which are flown in by quadrotors robots. To organize the quadrotors, a Vicon motion capture system is used to detect the positions of the quadrotor robots in the construction site, such that the structure can be assembled without the quadrotors colliding with each other. While the quadrotors have local control methods, such as PD controllers for attitude control to maintain consistent flight patterns, they ultimately rely on a centralized controller and its global vision data to follow the construction plan for the structure. This reliance imposes limits on the effective range of the construction system, by constraining the robots and structure to the accurate sensing range of the Vicon motion capture cameras.

The work of Augugliaro *et al.* [2] uses quadrotors as well to build 3D structures using foam bricks. The quadrotors are organized using centralized Vicon motion capture data, with markers placed on important locations such as the quadrotors, block pickup zone, and charging zone to locate each element within the construction site. A blueprint of construction instructions is followed by the centralized controller, which delegates construction tasks to the quadrotors and organizes their efforts in building the structure by generating the flight trajectories to follow. The controller also keeps track of the battery levels of the quadrotors, to be able to send them to

the charging station when low, thus enabling long continuous build times. While the system provides many useful capabilities such as self-organized charging, the reliance on global vision data limits the size of the construction site to the camera system's field of vision.

2.3 Building Material-Based Systems

The TERMES system [26] uses multiple manipulator robots that build 3D structures using a global plan by navigating the structure that they have built. The robots can guarantee the successful placement of objects due to being able to navigate the structure they build using marked custom build material, thus removing any terrain inconsistencies or environmental impasses. The robots orient themselves on the structure using the cross shape of white lines on each block that are sensed by sensors on the underside of each robot. The robots can locally sense where blocks have been placed, thus allowing the robots to detect where new blocks should be placed. Each robot can also locally sense other nearby robots, thus preventing collisions and allowing them to obey the traffic rules set by the compiler in the construction plan.

The system implemented by Stuart *et al.* [24] uses multiple multicopters to assemble structures using nylon fibers reminiscent of spiderwebs. By using a continuous material such as fibers, structures such as bridges can be built between gaps.

Grushin *et al.*'s system [13] uses the process of self-assembly to build structures, using the robots themselves as the building material. A seed block is used as the center of the structure, which specifies the construction plan, thus also implementing stigmergic construction. The blocks that attach themselves to the structure read the memory of the blocks they attach to, providing the information required to

orient themselves in such a way that they contribute to the overall final design of the structure.

2.4 Robotic Scaffolding Systems

A method of assembling structures using intelligent scaffolding to guide inert building blocks was developed by Komendera *et al.* in [16]. While the system is implemented in simulation as an algorithm instead of by using mobile manipulator robots, it shows a method for assembly that could be implemented using physical robots as an extension. The intelligent scaffolds have sensors and can move, make calculations, and communicate with each other. The scaffolds assemble structures by having a building material block be attached to one of the scaffold blocks in the scaffold group, causing the group to reconfigure to guide the build material to its destination. This process is repeated until the structure is completed, enabling large structures to be built using only three scaffolding blocks to guide construction. The construction process is determined by interpreting a desired structure into a list of actions that the scaffolds will execute to produce the intended results.

2.5 Summary

The main sources of inspiration for our system are the SRoCS [1] and TERMES [26] systems, which leverage stigmergy in different ways to build structures. Our intelligent scaffolding block system is based on SRoCS's stigmergic blocks, by having a seed block communicate with the worker robots, while simplifying navigation by using the navigable structure methods of TERMES. The structure of our navigable blocks (both intelligent and build material) uses the same indicative design features as the TERMES system, by having a white cross of lines on a black surface for the

robot to use to determine its orientation on the block. Instead of having the plan for the structure in memory on each robot, with a set of traffic rules to prevent collisions like in TERMES, we offload our plan to the intelligent scaffolding block, similar to SRoCS's implementation, thus allowing the plan to be provided to the worker robots in multiple segments. By segmenting the plan through the intelligent scaffolding block, the robot only holds a piece of the structure plan at a time, which also enables the opportunity to build the structure based on feedback using interrupts, or having the intelligent scaffolding block organize multiple robots to cooperate in building the structure. Thus, we combine elements of both systems to implement a new system.

Our system of intelligent scaffolding blocks more closely resembles the roles filled by the stigmergic blocks in SRoCS than the intelligent scaffolding by Komendera *et al.* [16], however we still take some inspiration from their work. Similar to how a structure is produced by converting the desired structure's representation into a list of actions for the scaffolds to execute, we take a representation of the final structure the user intends to create and convert it into a list of commands that the manipulator robots must follow to produce the structure. However, our implementation uses the intelligent scaffolding to send the assembly instructions to the manipulator robots instead of using the plan to move the scaffolding to produce the structure. Komendera et al.'s work also requires the scaffolding blocks to be physically connected in a small group, while our system keeps scaffolding blocks separate from each other, at various locations in the structure to provide a form of monitoring over a small area through the electrical connections between them.

The centralized control systems used by the quadrotor construction systems of Lindsey *et al.* [17] and Augugliaro *et al.* [2] were not considered as viable methods for organizing the worker robots in our CRC method because they violate the swarm

robotics tenet of only using local information by using global vision data and global communication methods.

For the construction goals of our thesis, a continuous build material such as the fibers used in Stuart *et al.* or the foam in Napp *et al.* [18] could not be used as they would not provide a way for the intelligent scaffolding blocks of our system to communicate with each other and would be unable to guide the navigation of the worker robots.

We also do not consider self-assembly for our collective construction system, as it violates the condition of removable scaffolding in our problem statement; with self-assembly our robots would become the structure itself. However we do implement a method of the blocks in the structure communicating with each other in our intelligent scaffolding blocks, similarly to the work by Grushin *et al.* [13].

Chapter 3

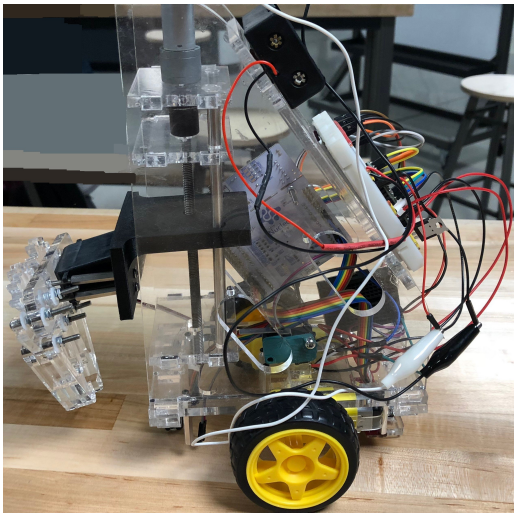
Methodology

3.1 Previous Work

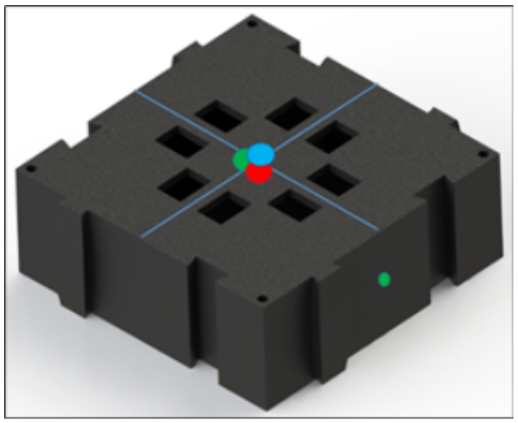
3.1.1 First Prototype (2017-2018)

The first prototype for this construction solution was developed in 2017-2018 by Cormier *et al.* [8]. They developed the first system of intelligent scaffolding blocks that communicated with each other through CAN (Controller Area Network) connections and commanded a manipulator worker robot. Their robot, shown in Fig. 3.1a, could successfully navigate along the scaffolding blocks by line-following. The robot would receive commands using a color sensor to sense color-coded commands from an RGB LED on the intelligent scaffolding block. However, their solution used large, fully 3D-printed blocks that were difficult to manipulate and connect, as shown in Fig. 3.1b. The gripper that the robot used required high accuracy, as well as holes for latching onto on the surface of the scaffolding blocks, which would cause the caster wheels to get stuck.

They also produced a construction algorithm in which the user would specify a structure, and the algorithm would produce the construction instructions for the



(a) Worker robot prototype



(b) Intelligent scaffolding block prototype

Figure 3.1: First system prototypes (2017-2018)

intelligent scaffolding blocks and worker robot. Cormier *et al.* decided to divide the structure into a spine row with reach columns that would provide consistent access to the entire structure for the robot as it navigated to place objects. A screenshot of the algorithm in simulation is shown in Fig. 3.2, showing the effectiveness of the solution given the successful operation of the physical robot and blocks. This algorithm narrows down construction into sequences of basic commands, such as 90° turns and moving forward one block, thus we developed our improved solution in Section 3.1.2. with regards to how the physical robot improvements could be designed to effectively use this algorithm. However, for our demos we did not explicitly use the algorithm from Cormier *et al.* because we only had five total blocks manufactured to build with.

3.1.2 Second Prototype (2018-2019)

The project was continued by Enyedy *et al.* in 2018-2019 [10]. We were tasked with improving the mechanical capabilities of the system, to increase the accuracy of

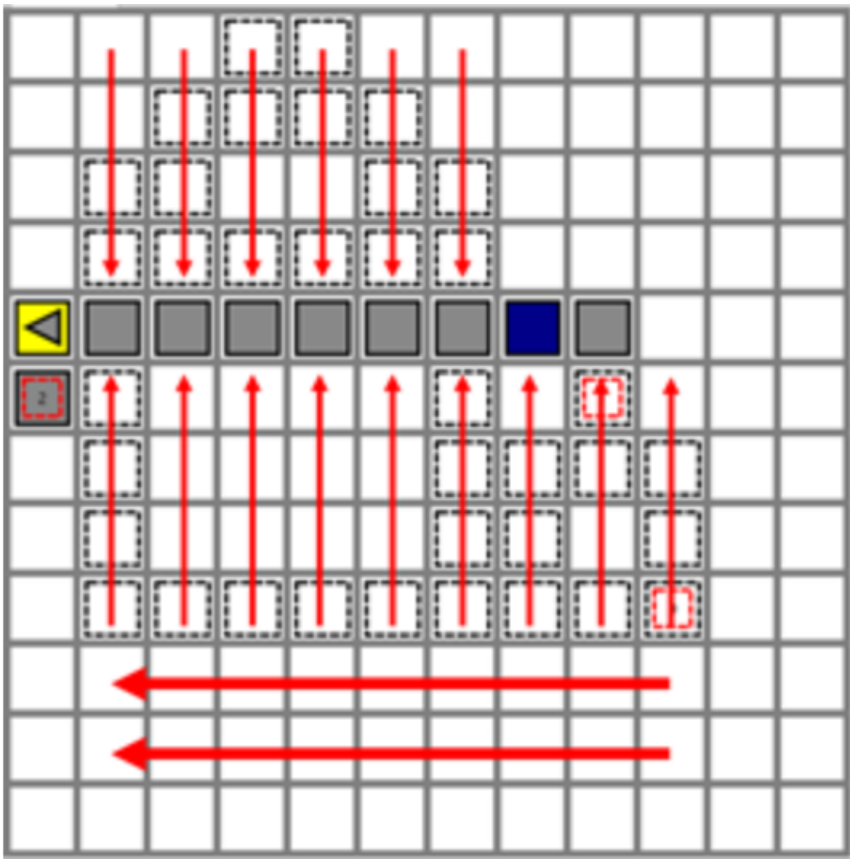


Figure 3.2: Construction algorithm in simulation

manipulating the blocks. We decided to redesign both the blocks and the robots from the ground up, keeping the most effective solutions previously implemented, such as the line-following navigation, RGB color commands, and the electrical connections between blocks for communication between intelligent scaffolding blocks. The robot was redesigned to have a center of mass closer to the ground, as well as a permanent magnet manipulator on the end of a four-bar linkage to simplify manipulating blocks, as shown in Fig. 3.4a. To detach the magnet from the magnet on the blocks, a linear actuator was used to physically separate the two magnets, as it was deemed simpler to implement than an electromagnetic end-effector. The new four-bar linkage CAD model and linear actuator are shown in more detail in Fig. 3.3, with the red portion of the linear actuator image showing the magnet mount for the end-effector, which

is attached at the end of the linear actuator to physically separate the end-effector from the blocks.

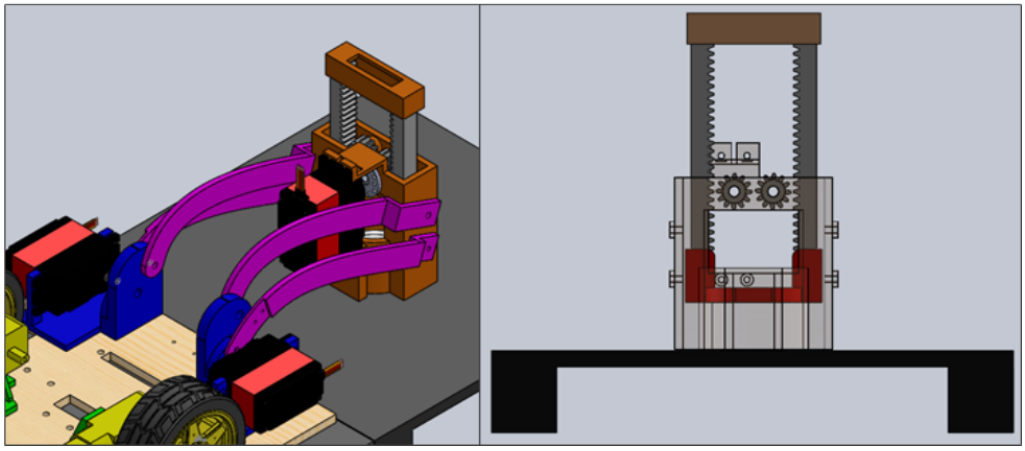
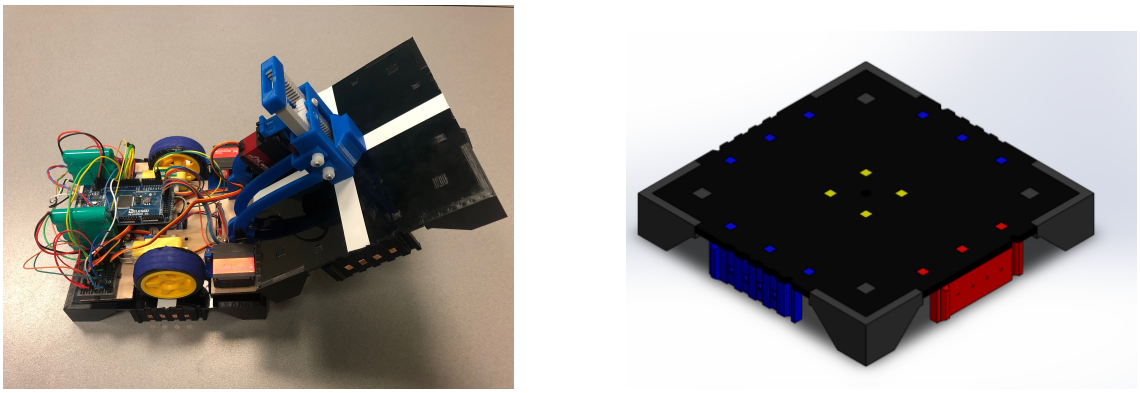


Figure 3.3: CAD models of four-bar linkage (left) and linear actuator (right)

Direction was added to the blocks' electrical connections, as shown in Fig. 3.4b, by having a single "male" connection side (using Pogo pins, shown in red in 3.4b) for input signals, and three "female" sides (using copper plates, shown in blue in 3.4b) for output signals. Instead of CAN, I2C was used for intelligent scaffolding block communication, with great success. To auto-correct block placement and ensure secure electrical connections, small magnets were placed in the corners of each block.



(a) Worker robot prototype

(b) Intelligent scaffolding block prototype

Figure 3.4: Improved system prototypes (2018-2019)

However, this solution required further improvements. The robot could manipulate blocks with a high success rate of 19/20 pick-and-place trials [10], however its navigation methods were no longer as accurate. The wheels were too slippery and the motors were too weak for the small corrections required for consistent, accurate navigation across multiple blocks. Due to the modular nature of the system, we could easily use it as a base for implementing the next iteration of the design.

3.2 Requirements

To successfully implement the proposed intelligent scaffolding-based construction system proposed, some improvements would have to be made to the previous implementation. While the work by Enyedy *et al.* improved the physical components of the work by Cormier et al., some of the navigation capabilities required adjustment to fully realise the minimum viable prototype of the intelligent scaffolding construction system.

To successfully navigate, higher torque and gear ratio motors would be required. The motors originally were geared too low to make small corrections, and once the robot picked up a block, they were too weak allow the robot to move between blocks. The tires of the wheels on the robot would have to be improved as well, as the lack of traction in [10] threw off the navigation. While the differential drive system originally used could be replaced by a holonomic system with omni- or Mecanum wheels, they were deemed too complex and unsuited for the slippery acrylic surface of the blocks.

In addition to higher torque and gear ratio motors, improvements to the navigation algorithms would be required to successfully navigate between blocks. To satisfy the conditions created by the low-torque high-speed motors originally used,

the navigation methods would overshoot many of the turns and had no time to correct movement between blocks. Thus, new algorithms for navigation would be developed.

The manipulation method would require an increased margin of error, to ensure successful block placement in the event of a slight misalignment. While the permanent magnet manipulator has a high margin of error when picking up blocks, placing the blocks requires correction based on the magnetic force attracting the sides of the blocks to each other.

The RGB color-command communication method of sending sequences of colors to represent commands worked well in concept, as the robot could successfully read all the color commands when directly on top of the center of the seed block. However, if the robot was a few millimeters off-center, which is an acceptable margin of error for manipulation, there would be a high level of noise in the HSV (Hue-Saturation-Value) color readings. The high level of noise decreases the effectiveness of this communication method in practice, thus an improved, more robust communication method would be required.

Finally, a new interpreter for user-designed structures would require implementation, as the new robots followed strings of integer commands. The interpreter would have to take a 2D schematic of the user's desired structure, as opposed to 3D due to the limitations of the robot and construction algorithm. then the interpreter must convert it into a string sequence of integers for each step of the plan to make the structure, with proper framework for sending and separating commands as required by the new communication method.

3.3 Navigation Improvements

3.3.1 Hardware

Improvements to the navigation algorithms were attempted using the original drive base motors (DAGU Hobby Gearmotors [9]). However, it was found that those motors supplied insufficient torque, at 800 gf-cm (0.0784 N·m), compared to what was required for the task. To calculate the torque requirements for the task, the largest total block mass from Fig. 18 of [10] was used (453 g, rounded up to 0.5 kg), as the blocks had not changed significantly in mass since the previous project and any extra magnet additions would only increase the mass by 5 g per magnet. The worker robot's mass was estimated to be around 1 kg, based on the mass of the heaviest components (batteries total 0.3 kg, 3 servos total of 0.2 kg) combined with an extra factor of safety to account for the 3D printed parts, wood, magnets, and wheels. The original wheels were 60 mm diameter, so we designed the chassis to accommodate wheels of roughly that size and based the radius of the torque calculation on the assumption we would choose wheels of the same size. Thus, the torque calculations determined the required torque for the motor to move the robot while loaded with an intelligent scaffolding block (max load). Using the torque equation, shown in Equation 3.1, we input our system's parameters to calculate that we require 45 kg·mm of motor torque to successfully move the robot, as shown in Equation 3.2.

$$\tau = mgr \tag{3.1}$$

$$\tau = 1.5\,\text{kg} * 9.8\,\text{m/s}^2 * 0.03\,\text{m} = 0.441\,\text{N} \cdot \text{m} \approx 45\,\text{kg} \cdot \text{mm} \tag{3.2}$$

Based on this result, a new motor that would fit the small amount of free space on the chassis was sought out. We selected a new motor from the Pololu Gearmotor

series [22] with the proper torque and gear ratio. Based on the small amount of space remaining on the chassis after the sensors and four-bar linkage were attached, the Pololu 20D Metal Gearmotor [22] series (6 V version) was selected for its ratio of size to torque, and based on our torque calculation, the 156:1 gear ratio was selected due to its relatively low maximum speed at 91 rpm and its high torque at stall of 79 kg·mm which was well within the limits required for the worker robot's navigation. Due to our working requirements of requiring roughly 45 kg·mm at max load on the robot, the motor would operate at roughly 18% efficiency, with a speed of 38 rpm, a current draw of roughly 1.6 A and almost at max power of roughly 1.75 W, thus operating well within the requirements of our system (such as battery current limits and sensor operation movement speed concerns).

The original 60mm diameter wheels that came with the DAGU motors also had low coefficients of friction, making it difficult to make the accurate 90° turns required for successful construction. Thus, we found Pololu wheels [23] of the same diameter, designed to fit the new motors, which have higher coefficients of friction to grip the slippery acrylic surface of the blocks. The new 60 mm x 8 mm wheels had tires that could grip the blocks' surfaces well, such that small turn adjustments could be made when performing 90° turns and for correcting movement while line-following between blocks.

3.3.2 Software

Once the drive base successfully made small position adjustments as required for correcting paths between blocks with only 8-inches of space between them, improvements to the navigation algorithms could be made. Initially, only the 6-sensor segments of our IR sensor arrays—QTR-8RC Reflectance Sensor Arrays [21] broken into 6-sensor and 2-sensor parts—across both the front and back of the robot were

used, shown in Fig. 3.5.

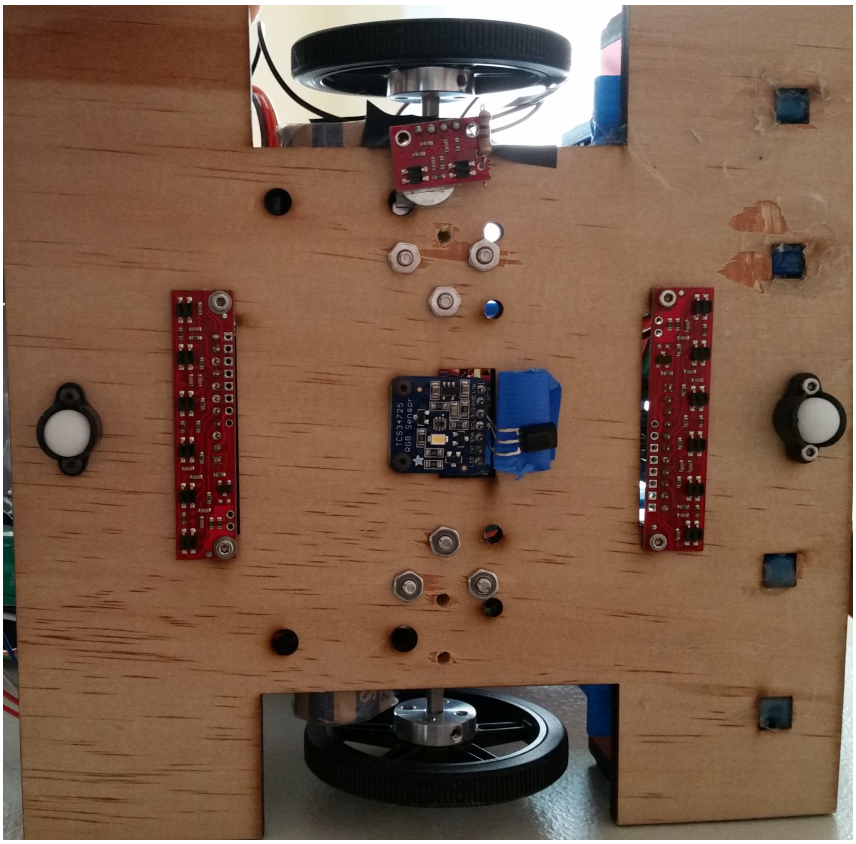


Figure 3.5: 12 light sensor setup in two arrays of 6

This setup of IR sensor arrays was sufficient for quickly sensing errors in the line-following path when traversing across multiple blocks, as shown in the testing platform in Fig. 3.6.

This testing platform was used to test various control methods for line following across the blocks. As only a set of seven unique errors and corrections are sensed during navigation, instead of using a full PID controller (proportional-integral-derivative controller) for line-following, which would require unnecessary extra computation and determines control based on continuous error sets, an error-correction lookup table for our discrete set of errors was implemented. To keep the robot on the line when traveling straight between blocks, a discrete set of errors and solutions

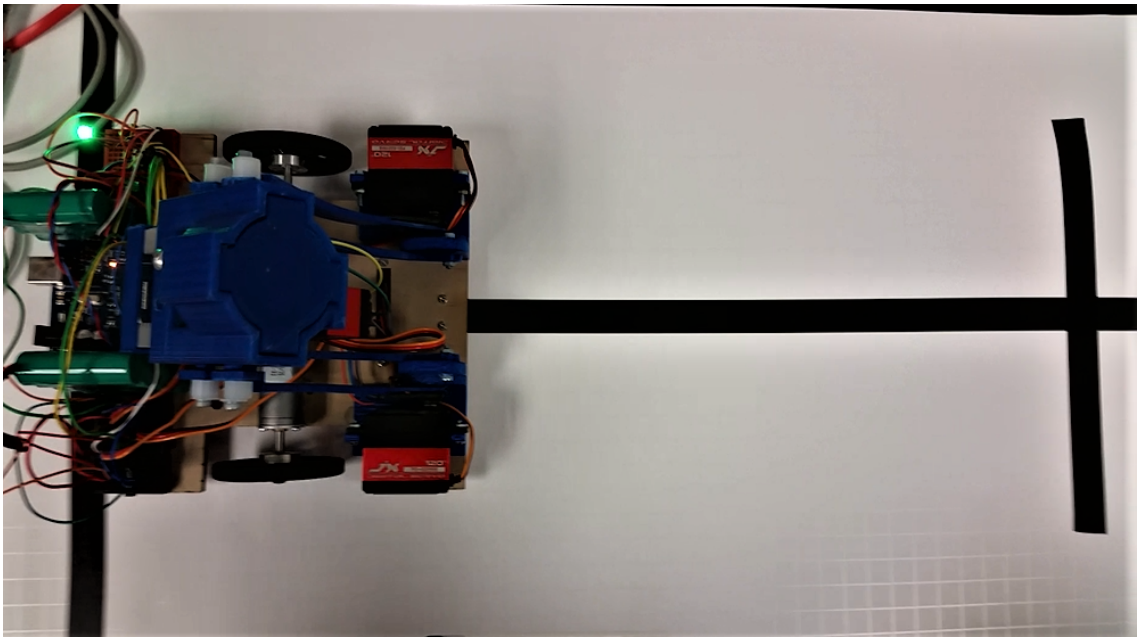


Figure 3.6: Line following test pad for developing error-correction Lookup Table

were devised. The line location was determined based on an exponential increase in values between the sensors' line detection output values, as shown in Table 3.1.

Table 3.1: Line location sensing setup

Front

0	1	2	3	4	5
-4000	-2000	-1000	1000	2000	4000

Back

5	4	3	2	1	0
4000	2000	1000	-1000	-2000	-4000

This setup of exponential increases in line location value increases the accuracy of the navigation system by greatly increasing the degree of error based on how far away from the center of the sensor the line is. For example, if the robot senses the line between sensors 3 and 4 on both the front and back line sensors, the line location is detected as 1500 (average between 1000 and 2000) and the error is calculated as

-3000 (-1500 error for both sensors, then combined). However, if the robot senses the line between sensors 4 and 5 on both sensors, then the line location is detected as 3000, and the error is calculated as -6000. Thus, a wider range of errors based on the line's detected location can be corrected for in the lookup table shown in Table 3.2.

The reversed order of line location values between the front and back line following sensor is used to determine whether errors are translational or rotational. If the front and back sensor both read the same sign (positive or negative) error, then the error is rotational, but if the front and back sensor have opposite signed errors, then the error is translational. Purely rotational errors are corrected by opposite direction rotations, and translational errors are corrected using different motor control inputs on each wheel to turn in an arc. Error correction is performed according to the lookup table shown in Table 3.2 by modifying the PWM of the motors based on a percentage of the base motor control signal of 70 PWM.

This method enables the robot to correct itself while moving forward, resulting in smooth motion between blocks as opposed to pausing to rotate. When the error is between $|4000-8000|$ then the robot makes a turn-in-place motion, because the error is high enough that it is a significant rotational error as opposed to a translational one. In the special case that white line is detected by all six sensors on the array, an error value of 10000 is returned, indicating to the robot that it has reached the next block and it should move to the center of that block.

However, using only the front and back light sensor arrays did not provide the required sensing capabilities to detect when the robot would be in the center of its destination block, a requirement for successful manipulation. Thus, attempts were made to detect the center of a block by locating the hole in the center of the block with the color sensor, which was ultimately deemed insufficiently accurate for the

Table 3.2: Line following error correction lookup table

Error	vR (%)	vL (%)
< -4000	80	-80
-4000	100	40
-3000	100	60
-2333	100	75
-1500	100	85
-666	100	90
0	100	100
666	90	100
1500	85	100
2333	75	100
3000	60	100
4000	40	100
> 4000	-80	80

task as the color sensor would have noisy readings and had difficulty picking up the centerhole of the blocks while the robot would be driving. Thus, another light sensor array was added on the centerline of the wheel axis, under the right wheel as shown in Fig. 3.5. This light sensor would be used to detect the midline of the destination block after the robot reached the next block according to the navigation algorithm, by having the robot resume its lookup table-based navigation until the midline sensors saw the white line in between themselves. Thus, sufficient sensing for navigating between blocks was achieved.

The method for rotating 90° was altered based on the new error calculation method, refining the turning process and increasing its accuracy. A similar lookup table method was implemented, but tuned for turning instead, as shown in Table 3.3.

Table 3.3: Line following error correction lookup table

Error	vR (%)	vL (%)
< -4000	100	-100
-4000	90	-90
-3000	85	-85
-2333	80	-80
-1500	75	-75
-666	70	-70
0	0	0
666	-70	70
1500	-75	75
2333	-80	80
3000	-85	85
4000	-90	90
> 4000	-100	100

As the main source of correctable error in the 90° turning algorithms is rotational, the turning correction lookup table is based only on changing motor PWM to adjust the rotation of the robot. The robot slows down when it nears its destination to ensure the sensors catch the line and stop the robot at the correct orientation.

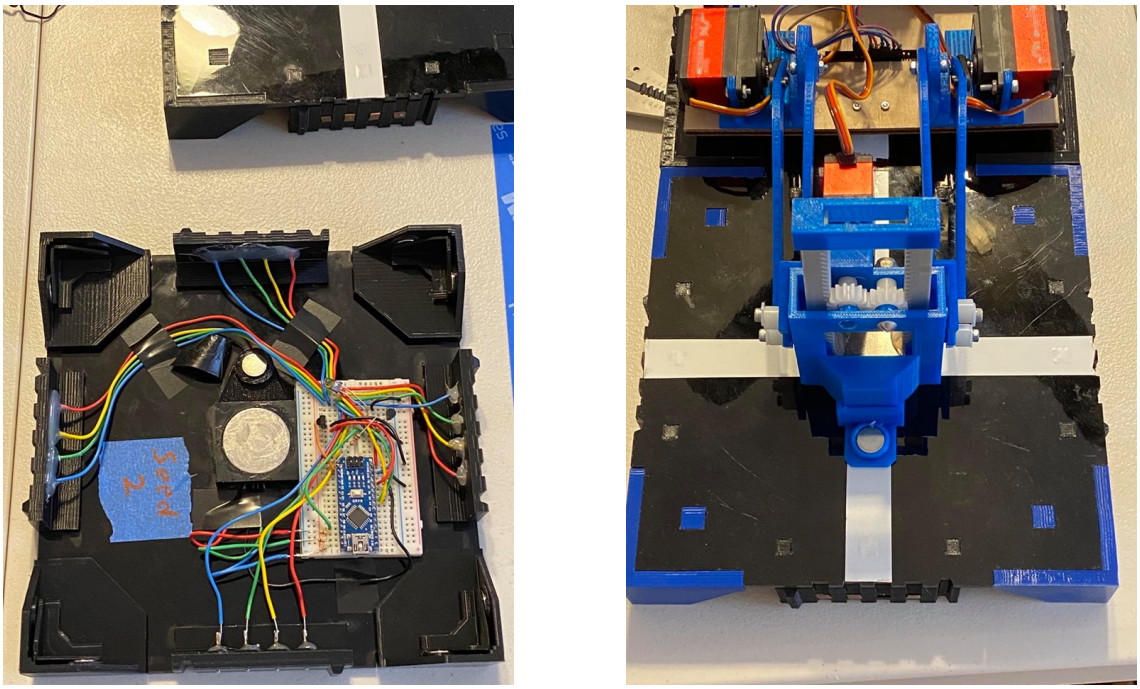
Overall the error lookup table method allows for a simple method for error correction that provides similar accuracy that a full PID controller would for our application while remaining computationally-inexpensive.

3.4 Manipulation Improvements

While testing the robot's navigation, it was found that despite the improved navigation and sensing capabilities, the turns still have inaccuracies. In the event of turning slightly more or less than 90-degrees, the block would end up placed wrong, whether on top of block the robot currently was positioned on due to the block's corner getting caught, or placed slightly diagonally away from the desired position, resulting in the electrical connections not conducing. On occasion, especially for manipulating scaffolding blocks, the block would rotate roughly 45° on the magnetic end-effector, a rotational error of the held block too great for the corner magnets to correct during block placement.

To correct any slight turning errors, the magnets on each corner of the blocks were doubled from originally two per corner to four. Thus, when placing the blocks, the strength of the magnets' attraction would rotate the block on the arm to the correct position, and even the robot would be slightly adjusted once the block reached close enough to its goal position. To prevent the block from spinning while attached to the end-effector magnet after being picked up, support magnets were added next to the main magnet on each block, as shown on an intelligent scaffolding block in Fig.

3.7a, and a support magnet was added in the respective position next to the gripper shown in Fig. 3.7b.



(a) Block support magnet

(b) Robot support magnet

Figure 3.7: Modular staircase

The increased weight of the block due to doubling the amount of magnets and adding the support magnet is still within the torque limits of the drive base motors when the robot navigates while loaded, because the additional nine magnets only add an extra 45 g. Thus, the new required torque is 45.4 kg·mm (or 0.454 N·m), which is a negligible increase from the calculated torque requirement used when selecting the new motors. The arm servos could also support the weight increase for the same reasons.

Thus, the increased strength of the magnets and addition of support magnets for manipulation, combined with the high-accuracy of the navigation algorithms, increases the robot's success rate for correct placement of blocks.

3.5 Communication Improvements

Once the robot could complete a hard-coded demo of picking and placing blocks in the desired location, without using instructions provided by the seed block, testing began on the effectiveness of using RGB-LED communication. The color sensor from Enyedy *et al.* [10] was initially still used, in the assumption that having the RGB LED on the intelligent scaffolding block change colors in color sequences representing the command sequences would be accurately sensed by the robot. However, it was discovered that the color-sensing capabilities of the robot were only accurate in the event of the robot's color sensor being positioned perfectly above the center of the seed block, which was the extent to which the color sensor communication was tested. In practice, the robot would only be aligned so perfectly at the start of the construction process, when the seed block delivers the first command to the robot. Once the robot executed the first block placement command and returned to the seed block for further instructions and to pick up a new block, it would no longer be so perfectly aligned. The high accuracy required by the sensor to see the correct colors resulted in a high quantity of noise being recorded when reading the color sequence commands after having placed the first block.

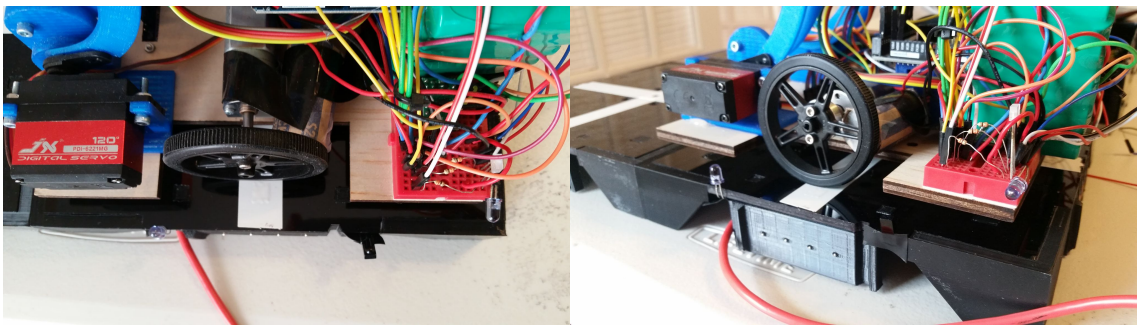
Alternative local communication methods between the blocks and the robot were considered. RFID was briefly examined, but was ultimately not tested due to complexity and cost. Infrared (IR) serial communication was suggested, functioning similar to a television remote changing channels on a television. IR was tested as the most viable alternative, due to its simplicity, low cost, and similarity to the original communication method, thus requiring minimum alterations to the hardware of the robot and blocks.

To establish communication between the blocks and the robot, an IR LED would

send IR signals to a TSOP38238 IR receiver [25]. The receiver only reads signals that are sent at 38 kHz frequency, thus the IR LED [11] requires a transistor circuit and connection to one of the PWM ports on the Arduino Mega to modulate a signal of such a high frequency. The data sent by the IR LED is through serial communication, allowing signals to be sent through serial print statements by simply connecting the Arduino Mega's serial-out (TX) port to the transistor circuit of the IR LED. The TSOP38238 receiver requires no other components other than wires, allowing it to be attached to the scaffolding blocks and the robots while occupying minimal space on the already crowded breadboards. It makes use of the RX serial-in port on the Arduino Mega, requiring the user to unplug the TSOP38238 each time they would like to upload a new program to the Arduino Mega, since uploading programs also requires use of the RX port. Thus, ensuring easy access to the wires was required to successfully implement the IR system as well.

It was found that with the original RGB LED color command system, the intelligent scaffolding blocks would not have a method for knowing when the robot was on top of them; a method for detecting when to send the next set of commands for the robot was required. Thus, when implementing the IR communication setup, both the robot and intelligent scaffolding blocks had IR LEDs and IR receivers in a location in which they could directly communicate with each other, as shown in Fig. 3.8a and Fig. 3.8b.

To ensure a command sequence would be fully read by the robot's IR receiver, indicator characters in the command sequences were implemented, as well as a unique indicator the robot would send to request commands. Since one-digit integers were the simplest to send, and the set from 0-9 is sufficient for expressing all indicators and commands to be followed by the robot, only the numbers 0-9 were used. The values read by the IR receiver are ASCII characters, so each command is converted



(a) (b)

Figure 3.8: 2-way IR communication between block and robot with IR LEDs and TSOP sensor of scaffolding block shown

into an integer first, to be easily decoded for the robot's movement state, then appended to the string of commands stored in the robot's memory. The meaning of each command is shown in Table 3.4.

Table 3.4: Command string sequence character value meanings

Integer	Indicator	Algorithm Reference
0	start of sequence	START
1	turn right 90 degrees	RIGHT
2	turn left 90 degrees	LEFT
3	pickup block	PICKUP
4	put down block	PUTDOWN
5	go forward 1 block	FORWARD
6	end sequence	ENDSEQ
7	<unused>	<unused>
8	request commands	<unused>
9	end of transmission	<unused>

Communication begins with the robot requesting the command for the current block that must be placed in the construction plan by sending the character `8` over its IR LED. Once the intelligent scaffolding block receives the request, it starts sending

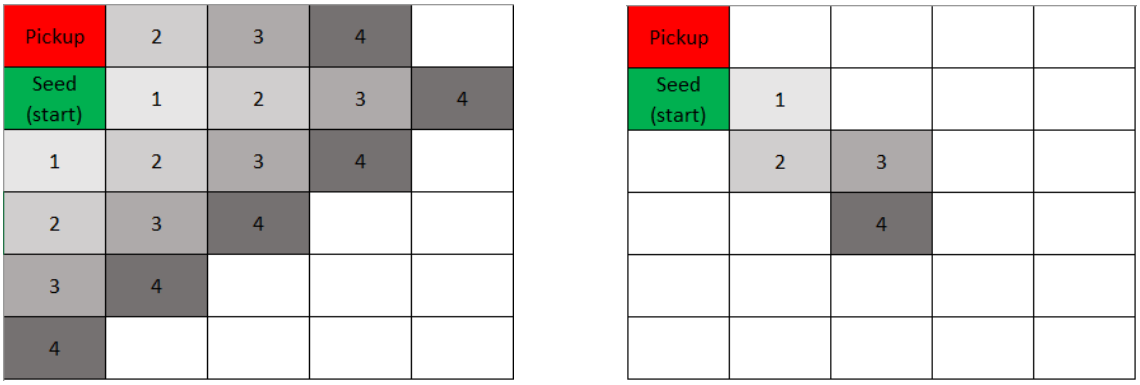
the command sequence, framed as "03<*navigation commands*>469" per command. The 0 at the beginning indicates to the robot that it should start recording the commands it is reading from the IR receiver and that it is not hearing the middle of a command sequence, thus ensuring that the commands are received and internalised in the proper order. Afterwards, the commands indicate that the robot would pick up the block (3), then navigate to the block's destination, and place the block (4). The 6 indicates to the robot that it has completed the command and can move to the next state of its building process, which is returning to the seed intelligent scaffolding block to pick up the next block. The 9 indicates to the robot that it has successfully recorded all commands in the command sequence and is not actually recorded in the command list stored in its memory, while simultaneously indicating to the intelligent scaffolding block that it has finished sending that set of instructions and should wait to send the next one until the robot has returned to request it.

3.6 Construction Algorithm

Since the robot receives commands from the scaffolding blocks and executes commands in the form of a string of integers representing framing bits and actual commands, an interpreter had to be developed that would convert the user-defined structure into a list of commands (represented by integers) that could be sent to the robot after being uploaded to the seed block.

While the work by Cormier *et al.* [8] developed a program to break a user-defined structure down into commands for the robot, the current implementation of our intelligent scaffolding system functions slightly differently, requiring a new interpreter design for users to define structures. Thus, starting at the beginning, the basic rules for reaching target locations were determined by drawing out by hand

multiple different configurations of structures the user could build and how those structures would be broken down into individual commands for the robot to follow. The paths for returning to the seed block after placing the blocks was also written out, to implement a function on the robot for reversing the command sequence in a manner that would return it to the home position. As the physical implementation of the system is limited to five total blocks (one seed block and four blocks to be manipulated), each test case for this algorithm's derivation was a different shape made by placing four blocks in a desired shape, as shown in Fig. 3.9b, with the entire set of tested four-block structures shown in Fig. 3.9a to check if the rules derived from the example held true across multiple structures and to check for any exceptions or edge cases. The numbers in the figures represent the order in which the robot would place the block. All of the commands required to build the structure in Fig. 3.9b and return to the seed block to pickup more blocks are derived in Fig. 3.10.



(a) All tested structures

(b) Example of a structure

Figure 3.9: Set of all cases tested, with an example case

As can be seen in Fig. 3.10, already a few patterns can be noticed from this single test case: each subsequent block that is placed uses the same command set to reach the previous block (plus a `FORWARD` command to reach that previous block position),

Place Block 1		Place Block 2		Place Block 3		Place Block 4	
Forward	Go Home	Forward	Go Home	Forward	Go Home	Forward	Go Home
PICKUP		PICKUP		PICKUP		PICKUP	
RIGHT	LEFT	RIGHT	RIGHT	RIGHT	LEFT	RIGHT	RIGHT
PUTDWN		FORWD	FORWD	FORWD	FORWD	FORWD	FORWD
		RIGHT	RIGHT	RIGHT	LEFT	RIGHT	RIGHT
		PUTDWN		FORWD	FORWD	FORWD	FORWD
				LEFT	RIGHT	LEFT	LEFT
				PUTDWN		FORWD	FORWD
						RIGHT	RIGHT
						PUTDWN	

Figure 3.10: Command list and returning to seed block command list for Fig. 3.9b

and then has an extra movement command appended afterwards followed by the manipulation command. Thus, the interpreter could simply design the forward command list one block at a time, appending a new command for each block just by checking the cardinal location of the block with respect to the previous block that has been placed. This simple relation led to the algorithm for the structure interpreter shown in Algorithm 1.

This structure interpreter was implemented as a Python script, using an input CSV in which the user specified the intended structure using the numbered locations as shown in Fig. 3.9b. The interpreter writes the command sequences to a text file, which then have the string of commands copied into the memory of the seed intelligent scaffolding block's memory. Then, the seed block splits the commands from each other in the long string of all commands, and send one command at a time to the robot using the IR communication system outlined in Section 3.5.

The reverse movement commands found in Fig. 3.10 were used to discover a set of rules for reversing the forward command sequence received from the block in the robot's memory for the robot to be able to find its own way home. Once

Algorithm 1 Structure Interpreter

```
1: previousPoint = ""; robotOrientation = "N"; lastLoc = 1, 0;
2: BM = 2D array representing structure
3: procedure FindBlock(desiredStep, startLoc)
4:     if BM[startLoc] == desiredStep then return error
5:     if BM[North of startLoc] == desiredStep then return "N"
6:     if BM[East of startLoc] == desiredStep then return "E"
7:     if BM[South of startLoc] == desiredStep then return "S"
8:     if BM[West of startLoc] == desiredStep then return "W"
9: procedure GeneratePlan(DO=FindBlock(desiredStep, startLoc),
   previousPoint, RO=robotOrientation)
10:    if DO == RO then return previousPoint
11:    if DO == RO + 90° Right then return previousPoint + "1"
12:    if DO == RO + 90° Left then return previousPoint + "2"
13:    if DO == RO + 180° then return previousPoint + "11"
14: desiredStep = 1
15: while desiredStep < number of blocks in structure do
16:    resultPlan = generatePlan(FindBlock(desiredStep, startLoc), previousPoint, robotOrientation)
17:    previousPoint = resultPlan + "5"
18:    append to commandlist file: "03" + resultPlan + "469"
```

the patterns for the home path were derived, such as whenever there is a 90° turn (`LEFT` or `RIGHT`) sandwiched between two commands for moving forward by one block (`FORWARD`), then reverse the direction of that turn to properly navigate back to the robot's starting position. The resulting home algorithm from this testing is shown in Algorithm 2.

Algorithm 2 Return to Seed Intelligent Scaffolding Block

```
1: home = remove PICKUP, PUTDWN and ENDSEQ commands from command sequence
2: if home.length() == 1 then
3:     if home.startsWith(LEFT) then
4:         home = RIGHT + ENDSEQ
5:     if home.startsWith(RIGHT) then
6:         home = LEFT + ENDSEQ
7:     return home
8: home = home reversed
9: n = 0 //index of command
10: if home.startsWith(FORWARD) then
11:     home = RIGHT + RIGHT + home
12:     n = 2 // skip checking the added right turns
13: while n < home.length() do
14:     if home[n] == FORWARD and n + 2 < home.length() then
15:         if home[n + 1] == LEFT then
16:             if home[n + 2] == LEFT then
17:                 home = home.substring(0, n+1)
18:                 n = n + 2
19:             if home[n + 2] == FORWARD then
20:                 home[n + 1] = RIGHT
21:                 n = n + 2
22:         if home[n + 1] == RIGHT then
23:             if home[n + 2] == RIGHT then
24:                 home = home.substring(0, n+1)
25:                 n = n + 2
26:             if home[n + 2] == FORWARD then
27:                 home[n+1] = LEFT
28:                 n = n + 2
29:     n++
30: return home + ENDSEQ
```

This algorithm is run once the robot has completed its task of placing the block

for the current command in its memory from the intelligent scaffolding block. The robot pauses for a moment and runs the return home command, which produces the modified reversed command sequence path that leads back to its starting position. Once the return home command list is formed, it is stored in the robot's memory, replacing the original sequence of movement commands, and is followed in the same manner that the robot followed the forward command list it received from the intelligent scaffolding block.

Chapter 4

Experimental Evaluation

4.1 Manipulation Tests

Once the new hardware upgrades had been completed, we performed tests to check the manipulation capabilities of the robot and see if they had changed given the dimensions and parameters of the new motors. In the initial tests, it was discovered that, when the robot would pick up a block, it would begin to tip forward and even fall forward in exceptional cases. With the previous motors, the robot was more balanced, as the previous motors had shafts at 90° output as opposed to the standard direct output shafts of the new motors, thus allowing most of the mass of the previous motors to be located closer to the back of the robot. The previous wheels also had greater mass than the new wheels, which are much thinner.

We determined that due to the new distribution of mass on the robot, a counterweight must be added to increase the mass of the back side of the robot, thus stabilizing the robot by shifting its center of mass. 100 g counterweights were added alongside each drive motor, towards the back side of the robot. With the counterweight in place, the center of mass remains within the support polygon of the

robot's wheelbase, thus keeping the robot stable during the pickup process. The extra 200 g of mass on the robot results in a new estimated total mass of the robot plus an intelligent scaffolding block of roughly 1.7 kg, resulting in a required motor torque to move the robot of 50 kg·mm, which is still within the safe output of the new motors.

4.1.1 Purpose

After adding the counterweights, manipulation tests were performed to check the reliability of the manipulation mechanism. These tests would show how stable the robot would be during manipulation, as well as what potential underlying causes for error exist during block pickup or placement.

4.1.2 Setup

The robot was placed on the seed block, as shown in Fig. 4.1, and we ran the manipulation code for picking up and placing the block 50 times in a row. The robot began with two full batteries—to ensure the ability to continuously test without waiting to charge a battery in the event of a power failure—and had to pick up one of the build material blocks that had more refined 3D printed parts, as the quality of the build material blocks varies. A short manipulation test using the lowest-quality build material block was completed as well, to test the impact of the block's hardware on manipulation.

4.1.3 Results

Out of the 50 trials, the robot successfully picked up the block 48 times and placed it successfully 47 times. Two of the pickup and place failures were due to battery

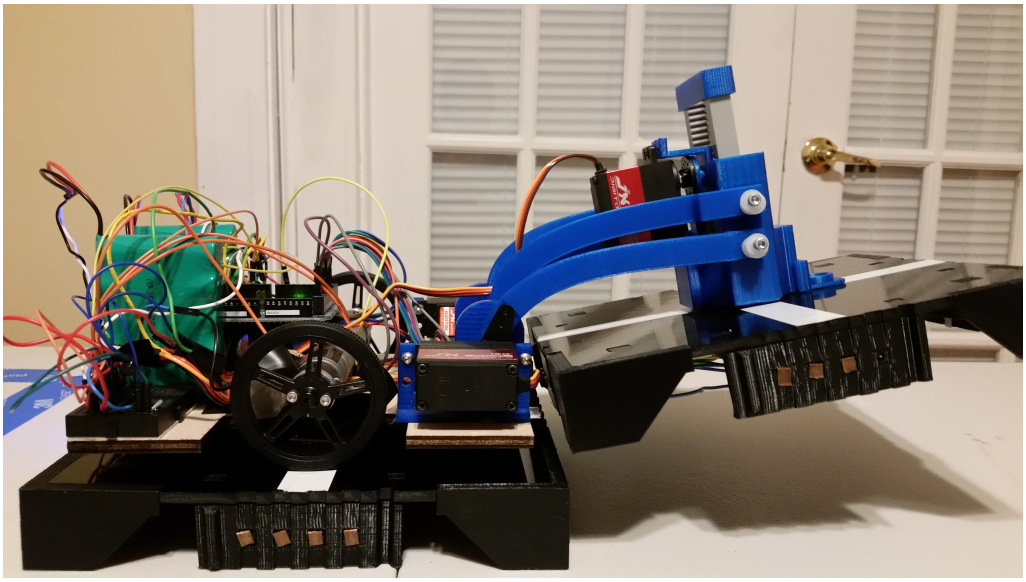


Figure 4.1: Block manipulation test setup

failure, however. The one extra placement failure was due to a misalignment of the robot at the start of the 50-block trial, thus once the robot was realigned it could pickup and place blocks with 100% accuracy (not counting the battery failures).

The trial was tested with one of the lower quality build material blocks as well, indicated by blue corner pieces that were 3D printed at a later date than the other blocks' black corner pieces. The lower quality of the block's 3D printed parts causes the block to catch on corners more easily, and the block's more slippery surface causes the block to spin a few degrees while being lifted by the robot. Thus, the lower quality block's support magnet was doubled to prevent it from spinning during manipulation. Once the support magnet's magnetic force was increased and the robot was realigned, the robot could pick up and place even the lower quality block with 100% accuracy (10 out of 10 trials).

Thus it was discovered that the main causes for errors with block manipulation are caused by translational or rotational misalignment such that the placed block's corner would catch on the previous block's edge and prevent accurate placement.

To improve the reliability of manipulation, improvements to the support magnet system that prevents the block from spinning would be required. The manufacture quality of the blocks would require improvement as well.

4.2 Navigation Tests

In these navigation tests, we tested the robot's ability to turn 90 degrees and the robot's ability to move between blocks. The robot's ability to move between blocks (inter-block navigation) was combined with more 180° turn tests, to ensure the robot could run in a continuous loop reflective of the navigation required in the full demo.

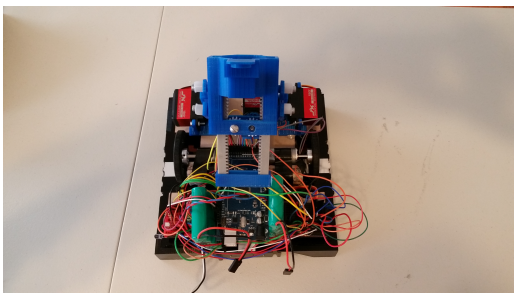
4.2.1 Turning Tests

Purpose

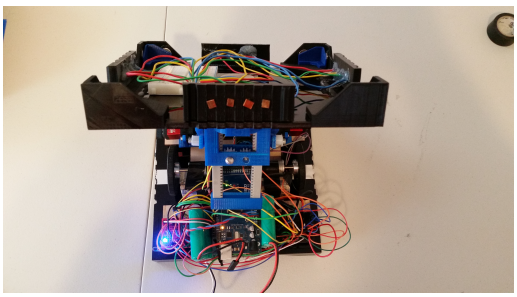
The turning tests seek to show the reliability of the robot's ability to perform 90° turns, the core component required for the robot's navigation and manipulation to succeed. Due to the differences in total mass of the robot when loaded and unloaded, trials of both were performed. Success in each trial was determined by whether the robot turned successfully within 5° of the goal of 90°, determined by the output of the RGB LED status indicator on the robot. These trials also aim to diagnose the main underlying causes of inaccurate turns by analyzing the repeated turns.

Setup

To test the turning capabilities of the robot, it was placed on a build material block facing north (away from the camera) as shown in Fig. 4.2a for the unloaded turning test and in Fig. 4.2b for the loaded turning test.



(a) Unloaded turn test setup



(b) Loaded turn test setup

Figure 4.2: Turning test starting positions

The robot would then perform two 90° left turns (to showcase capabilities of performing both 90° and 180° turns in one trial) and then two 90° right turns. The robot would loop these four turns 20 times. In the event that the robot would turn off-center enough such that all four turns fail (by having more than a few degrees of error), the robot's position would be reset and a failure would be recorded for all turns in that trial run.

Results

Out of the 20 unloaded turning tests, the robot ended up off-center severely enough to require a full position reset twice. The full position resets were caused by accumulating faults during the second right turn in the sequence, as shown in Table 4.1a, thus not reflecting the success rate of the other turns because despite the best efforts of any turn, a translational error caused by accumulating motion off-center cannot be corrected by a single, pure rotational motion. Thus, when discussing the accuracy of our turns, we will present their percentage success rate based on both the removal of the full reset trial data points and the raw trial data.

The first left turn in the sequence reached the goal of turning within a few degrees of 90° 16/20 times, with a success rate of 80% when counting the accumulated error caused by the second right turn. The success rate of the second left turn and the first

right turn in the sequence are both 18/20 (90%). These two turns reap the benefits of the correction made by the first left turn, even in the event that the first left turn is slightly off, and reflect the robot's ability to self-correct from small translational error. The second right turn in the sequence only succeeded 5/20 times (25%), and was the source of the accumulated translational error that caused the full resets.

The observed cause of the errors is accumulated translational error, causing the robot to be off-center on the block, misaligning the line following sensors with the white lines. This misalignment results in over-turning to sense the line in the desired position, causing the robot to end the turn with an orientation that prevents correct block placement.

	Left1	Left2	Right1	Right2
# of Success	16	18	18	5
# of Failure	4	2	2	15
Total:	20	20	20	20

(a) Unloaded turn test setup

	Left1	Left2	Right1	Right2
# of Success	20	17	20	14
# of Failure	0	3	0	6
Total:	20	20	20	20

(b) Loaded turn test setup

Table 4.1: Turning test results

Out of the 20 loaded tests, results of which are shown in Table 4.1b, the robot never required a full reset. The errors of the second right turn were much fewer than in the initial unloaded test, and could be attributed to light sensor calibration errors caused by the light source placement. The first left turn and first right turn both succeeded 20/20 times (100%), showing the high accuracy of 90° turns of the robot

even when loaded with a block. The second left turn only succeeded 17/20 times (85%) which can be attributed to the accumulation of slight translational error, as the second left turn only failed during the trials that the second right turn failed previously (see trial numbers 14-16). The second right turn succeeded only 14/20 times (70%), which is a vast improvement over the unloaded success rate, however it is still quite low. The additional mass from the block could explain the overall increase in accuracy and lack of full resets required, as the robot has higher friction and moves slightly slower which benefits the slight turn corrections and light sensing line detection.

The recorded "failures" in the turning tests are still within an acceptable rotational error range such that they can all be recovered from during the line-following movement between two blocks, as shown in the inter-block navigation tests in Section 4.2.2, but they have a high chance of causing failures during block placement, as the placed block will get caught on the edge of the previous block (preventing magnetic correction) if the robot has significant rotational error.

4.2.2 Inter-block Navigation Tests

Purpose

The inter-block navigation tests seek to show the robustness of the navigation system. As seen in the turning navigation tests, the robot occasionally makes turning errors, whether caused by environmental inconsistencies that affect the light sensors or by accumulated translational errors caused by the hardware of the system. Inter-block navigation offers the opportunity to correct those translational errors during the line-following movement between blocks; the robot self-corrects on the way to the next block. More importantly, this test shows the robot's effectiveness at

navigating between different lengths of straight lines of blocks. Testing on Z-shaped block structures is left to the full demo testing portion, as in this test we maintain focus on straight motion between blocks.

Setup

We set up multiple tracks of blocks for the robot to navigate across. 180° left turns will be performed at the start and end points of each trial, such that the trial can be run on a loop and such that the self-correcting behavior provided by the turns can be showcased.

We conduct the first trial navigating between two blocks 20 times, to show a unit test of inter-block motion, with a setup shown in Fig. 4.3. We conduct the second trial navigating between all five blocks available in this minimum viable prototype, as shown in Fig. 4.4, to test the maximum range of consecutive forward navigation commands possible with our limited number of blocks.

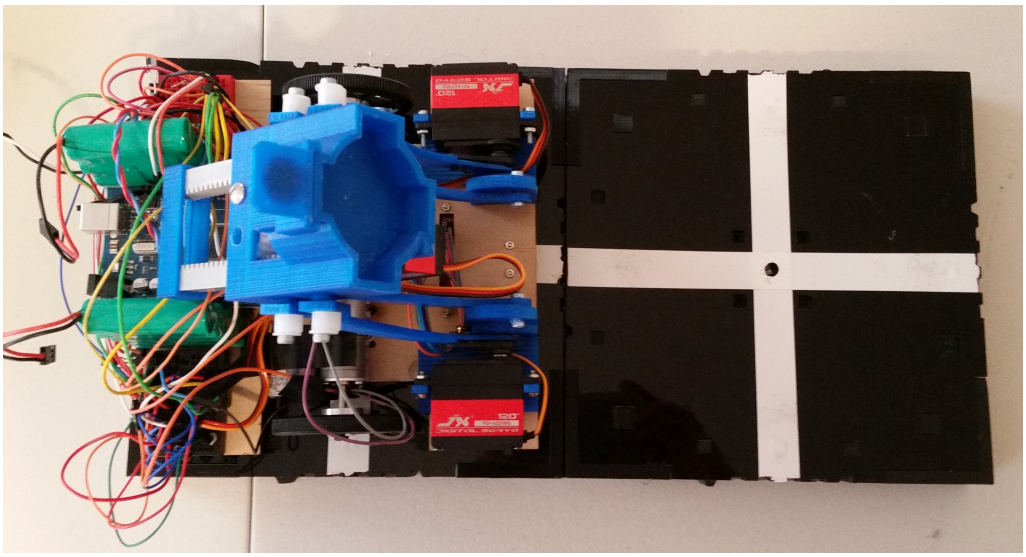


Figure 4.3: Two block inter-block navigation test

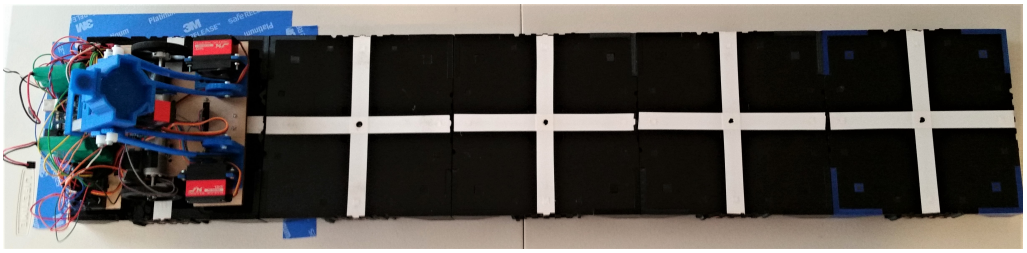


Figure 4.4: Five block inter-block navigation test

Results

The results from the two-block trial are shown in Table 4.2, and the results from the five-block trial are shown in Table 4.3. A successful trial consists of the robot successfully traversing the block (or completing the 180° turn within a few degrees of the goal), A half-successful trial (recorded as 0.5) indicates that the robot had difficulty traversing across one of the blocks, but still traversed it after struggling for a few seconds, and a failed trial consists of the robot being unable to traverse the block or completing the 180° turn within more than a few degrees of the goal.

Table 4.2: 2-block navigation unit test

	Block 1	Block 2	180 turn
# of Success	20	20	15
# of Failure	0	0	5
Total:	20	20	20

Table 4.3: 5-block navigation test (yellow column indicates lower-quality block)

	Block 1	Block 2	Block 3	Block 4	Block 5	180 turn
# of Success	20	20	20	20	17.5	16
# of Failure	0	0	0	0	2.5	4
Total:	20	20	20	20	20	20

Both trials show that the robot can consistently travel between blocks. In the

two-block trial, the robot successfully travels between the blocks 20/20 of the trails. The failures in the 180° turns are due to the robot turning slightly off-center, resulting in a turn more than a few degrees off of the desired orientation. However, as the results of the navigation test show, the robot corrects itself from those turning errors, and reaches the next block regardless.

In the five-block trial, the robot navigates successfully between all of the blocks consistently, except for when traveling between one of the higher-quality blocks and the lower-quality block with the blue corners. The block with the blue corners, represented in Table 4.3 as "Block 5", has some trials listed with a result of 0.5. That value indicates that while the robot could eventually traverse Block 5, it had difficulty doing so due to manufacturing errors of the block's 3D-printed parts. Due to these manufacturing errors, the block's surface is not exactly level with the other blocks' surfaces, causing the caster wheels to get caught. However, the robot slowly overcomes this height difference as it eventually gains enough traction while spinning its wheels.

4.3 Communication Tests

4.3.1 IR Unit Test

Purpose

This thesis was our first time using IR communication on a robotic system, thus a simple unit test to understand how to use the IR LEDs and receivers was set up before implementing them on the robot and intelligent scaffolding blocks.

Setup

The unit test setup is shown in Fig. 4.5, with the IR LED hooked up to an Arduino Uno with a BC547 transistor and resistors to ensure that the LED could send a 38 kHz signal for the IR receiver to read. Another Arduino had the TSOP38328 connected to it. The IR LED transmitted signals in a loop that would send the integers 1 through 100 on repeat.

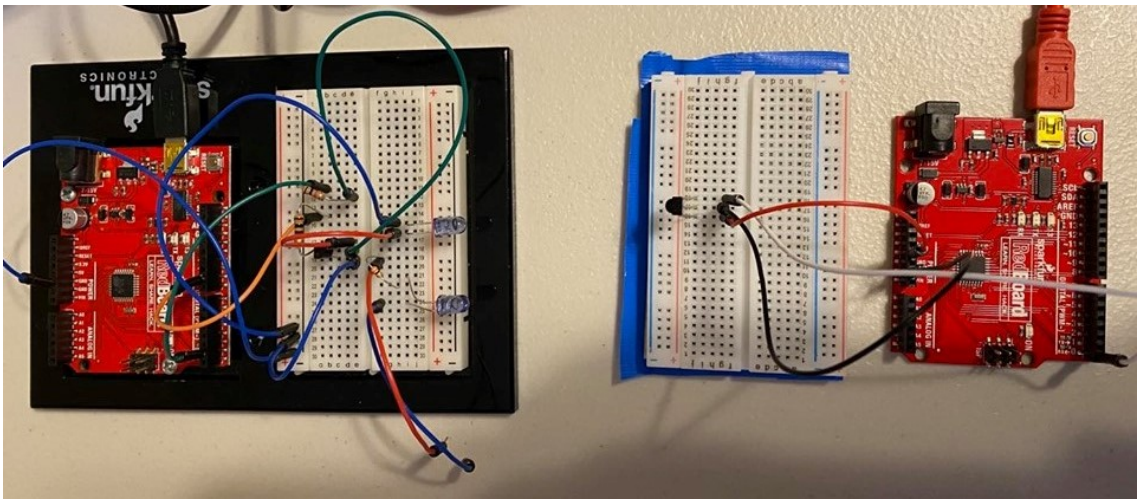


Figure 4.5: IR communication unit test setup of IR LED sending signals directly facing a TSOP38328 sensor

Results

The successful output of the integer data between 1-100 transferred between the IR LED Arduino and the IR receiver Arduino at 1200 baud is shown in Fig. 4.6.

It was found that the IR signal could travel around obstacles, as putting a hand or sheet of paper between the IR receiver and the IR LED would not interrupt the transmission. However, when a large, black-colored surface would be placed in between the LED and receiver, it was found that the signals could not be received; the IR signal was completely absorbed by the black surface. Due to the black acrylic

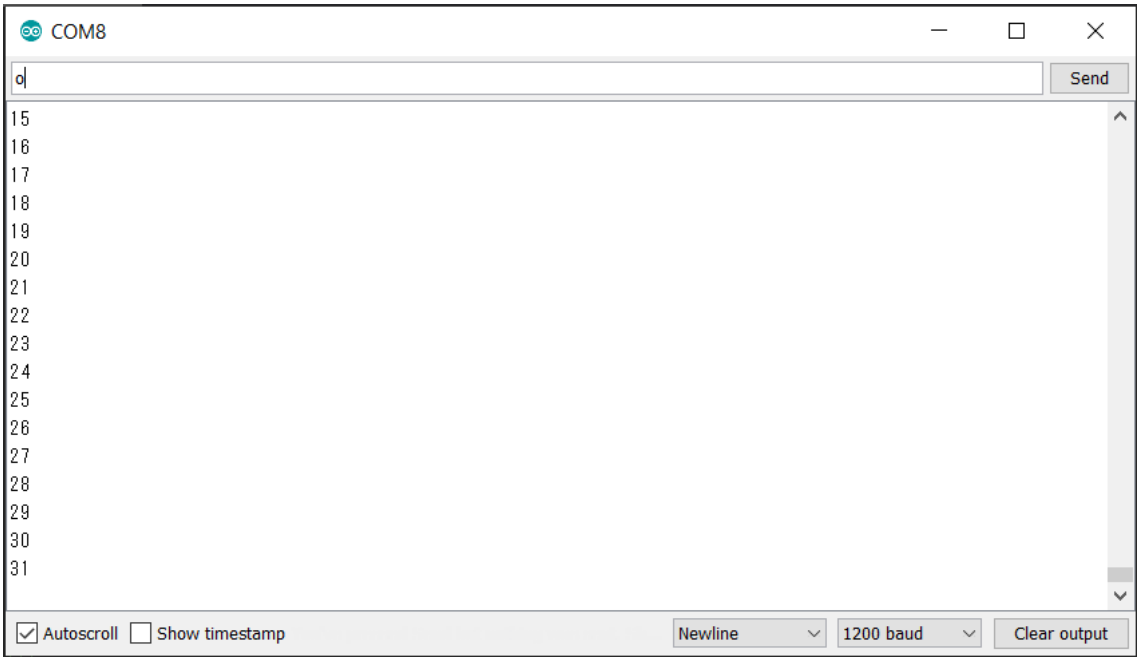


Figure 4.6: Results of IR communication unit test of integers received by TSOP38328, showing expected results of series of integers increasing from 1-100

surface of the blocks, the IR LEDs and receivers on the robot and intelligent scaffolding blocks would have to be in direct line-of-sight with each other and mounted above the acrylic surface to avoid any loss of messages. Thus, the IR communication system was considered reliable and robust enough to implement on the robot and the intelligent scaffolding blocks.

4.3.2 2-Way IR Communication Test

Purpose

To test the effectiveness of the IR communication system between the blocks and the robots, we implemented an IR receiver and LED on both the robot and the intelligent scaffolding block. We tested the range of the communication method, with respect to the locations that the sensors and LEDs could be placed on. Thus, we could determine where the best locations to place the IR components to ensure

reliable communication.

Setup: IR Underneath Block

Due to the ability to bounce the IR signals off of surfaces, the IR LED and receiver were initially placed on the underside of the intelligent scaffolding block, as the hole in the center of the block only had enough room to hold the RGB LED from the previous communication method. The intelligent scaffolding blocks' RGB LED and manipulation magnet holders were fused to the acrylic surface of the block, thus placing the IR LED and/or IR receiver in the hole in the center of the intelligent scaffolding blocks was impossible without significant modifications to the blocks. The IR receiver for the robot was placed on the robot's chassis' underside, right in front of the color sensor. Instead of having the robot execute the commands physically upon receiving them, the robot would flash a color sequence representing the commands instead.

Results: IR Underneath Block

The testing of this sensor placement proved to be unsuccessful, as there would be no way for the robot's IR LED (which would be added in the next iteration of the communication system) to send a signal that the intelligent scaffolding block would receive, as the black acrylic surface would absorb the signal before it could be received. Thus, we decided to mount the IR communication system on the robot and the intelligent scaffolding block such that each LED had direct line-of-sight to the corresponding receiver, as shown in the next test.

Setup: IR Above Block Surface

Upon discovering that the signal could not reach underneath the block reliably, the IR communication system setup implemented in Fig. 3.8b and Fig. 3.8a was developed. The IR signals would now have direct paths between the robot and the intelligent scaffolding block. The IR receiver was positioned slightly above the acrylic surface, because in testing it was found that if the IR receiver was mounted too close to the acrylic surface of the intelligent scaffolding block (i.e. resting on top of it), the IR signal from the robot would be completely absorbed before it could be recorded by the block. To test the effectiveness, the robot was put directly on top of the intelligent scaffolding block, aligned as it would be during the full demo tests in Section 4.6, such that the robot's IR LED would be pointing towards the intelligent scaffolding block's IR receiver. The robot would send its unique command request bit, then await the transmission of the sequence of commands from the intelligent scaffolding block. Again, instead of having the robot execute the commands physically upon receiving them, the robot would flash a color sequence representing the commands instead.

Results: IR Above Block Surface

The robot successfully requested the commands from the intelligent scaffolding block, and then await receiving the command list. The timing of the request and transmission worked as expected during the communication trial, such that the robot did not miss any of the bits of the command sequence and could successfully execute the commands after they were received, indicated by the robot flashing the expected color sequence upon completely receiving the build plan instructions.

4.4 Construction Interpreter & Home Algorithm

4.4.1 Construction Interpreter Tests

Purpose

Upon deriving the rules required to generate the construction interpreter algorithm, solutions to five structure configurations were available for testing the construction algorithm against. Thus, to prove that the construction interpreter algorithm successfully could develop plans for building any structure using five total blocks (including the seed block), we tested each possible configuration combination shown from Fig. 3.9a.

Setup

When deriving the construction interpreter algorithm, five test cases were used: a horizontal straight line, vertical straight line, vertical Z, horizontal Z, and horizontal L. Thus, the interpreter was run on these inputs and its output construction plan was compared against each hand-written plan. To specify the structure for this test case, the CSV file was edited as shown in Table 4.4.

Table 4.4: Structure specified in CSV, with -9 indicating seed block and -10 indicating block resupply depot

-10	0	0	0	0
-9	1	0	0	0
0	2	3	0	0
0	0	4	0	0
0	0	0	0	0
0	0	0	0	0

Results

Running the construction interpreter Python script produces the string of commands shown in Equation 4.1 that is then copied to the seed block's memory before building the structure. This string of commands is an integer representation of the individual commands required to build the entire vertical Z configuration example specified in Table 4.4, thus successfully producing the required construction plan as expected.

$$03146903151469031515246903151525146 9 \tag{4.1}$$

The commandline output for the Python script test case for the vertical Z block from Fig. 3.9b is shown in Fig. 4.7, which shows written out that the commands produced will result in an accurate construction plan. Each of the test case structures produced the expected resulting construction plans and command lists. After those five explicitly written out test cases, each possible structure outline by Fig. 3.9a was tested, which discovered an edge case that structures could not be properly interpreted if any blocks were on the same row as the resupply depot—the error was quickly caught and for the rest of the testing, the algorithm worked as expected with no erroneous edge cases when developing structures using only the four blocks available for our prototype. Thus, the interpreter was deemed effective enough to use for planning a full construction demo using five total blocks, shown later in Section 4.6 about full-demo testing.

4.4.2 Home Algorithm Tests

Purpose

To test the algorithm for reversing the forward command list received by the robot from the intelligent scaffolding block such that the robot could return to the seed

```
(base) D:¥!!!WPI¥!MastersThesis¥ConstructionInterpreter>python csv_reader_python.py
seeking block: 1
lastX, lastY before findBlock: 1 0
BM at[1][1] = 1
lastX, lastY after findBlock: 1 1
desired orientation: E vs current orientation: N
resultplan: 1
seeking block: 2
lastX, lastY before findBlock: 1 1
BM at[2][1] = 2
lastX, lastY after findBlock: 2 1
desired orientation: S vs current orientation: E
resultplan: 151
seeking block: 3
lastX, lastY before findBlock: 2 1
BM at[2][2] = 3
lastX, lastY after findBlock: 2 2
desired orientation: E vs current orientation: S
resultplan: 15152
seeking block: 4
lastX, lastY before findBlock: 2 2
BM at[3][2] = 4
lastX, lastY after findBlock: 3 2
desired orientation: S vs current orientation: E
resultplan: 1515251
```

Figure 4.7: Commandline output for construction interpreter on vertical Z block

block for its next instruction, a simple test was run on an Arduino Uno using serial print statements and a replica of the robot's state machine.

Setup

Instead of executing the commands, the Arduino Uno's serial print statements would indicate the robot's command following progress. The print statements allowed us to check if the return home algorithm would produce the correct path for the robot to return to the seed block.

Results

An example of one of the results of these tests is shown in Fig. 4.8, in which the robot would successfully reach the seed block based on the generated home command (as checked against the hand-written test cases). The home command tests were run on each structure configuration in the construction interpreter trials, and successfully

generated an effective list of commands to return the robot to the seed block.

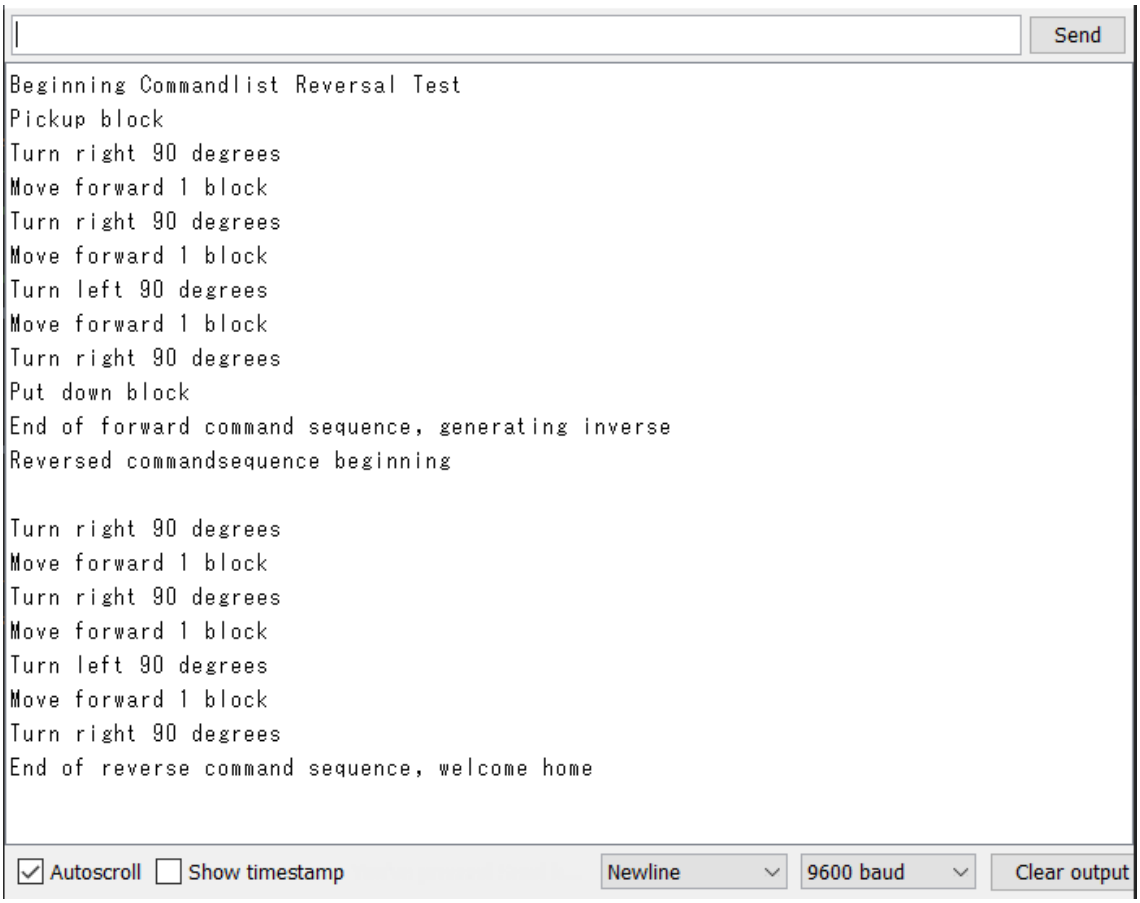


Figure 4.8: Arduino serial window showing results of home command unit test for placing the final block in the vertical Z block test from Fig. 3.9b

4.5 Block Communication (I2C) Testing

4.5.1 Purpose

The I2C communication between the intelligent scaffolding blocks worked successfully in Enyedy *et al.* [10], but we decided to test if the Pogo pin connections still worked as intended such that the blocks could still communicate with each other. This communication system allows for the seed intelligent scaffolding block to detect

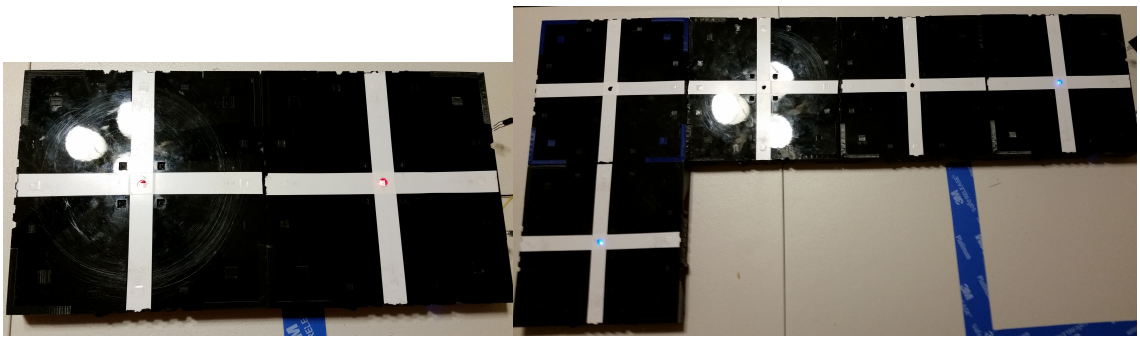
when other intelligent scaffolding blocks are added and removed from the structure, enabling it to monitor the status of the structure between it and the added scaffolding block.

4.5.2 Setup

Since the intelligent scaffolding blocks' hardware was already proven to work successfully, they were unchanged. Thus, this test was performed by uploading the I2C code onto both blocks from [10], and then powering the seed block and connecting various numbers of build material blocks in between the seed block and its leaf intelligent scaffolding block. If the Pogo pins and copper plates connect properly, then the leaf block's RGB LED lights up white. An 8.4V rechargeable battery was attached to the Vin and GND ports of the Arduino Nano of the seed block, to ensure there would be enough current to travel between the long sequences of wires between all of the blocks; in preliminary tests the seed block was powered by the USB cable, but it did not provide enough current for the signal to properly reach the leaf block. Once the leaf block is added to the structure, it requests the seed block to assign it a unique ID in the structure, such that it can be specifically communicated with. Once communication is properly established, the RGB LED on the both the seed block and the leaf block should begin flashing the same color sequences in sync with each other.

4.5.3 Results

Regardless of where the leaf block was placed, its RGB LED would light up and then begin to flash a color sequence in sync with the seed block, as can be seen in Fig. 4.9a and Fig. 4.9b. Thus, it was shown that the I2C connections still functioned properly and could be used in the full demo testing in section 4.6.



(a) I2C between 2 blocks (b) I2C between 5 blocks

Figure 4.9: I2C communication between blocks showing synchronized color sequences

4.6 Full Demo Test

4.6.1 Purpose

To show that all of the components of our collective robotic construction system function together as intended to build a desired 2D structure, we performed full demo tests. These full demo tests would show the effectiveness of our construction system as a whole and pinpoint the aspects of the system most prone to failure, to improve those aspects in future iterations.

4.6.2 Setup

We began by setting up the "construction site" as shown in Fig. 4.10, with the robot beginning on top of the seed intelligent scaffolding block facing north towards the block pickup zone. The other blocks for building the structure are stacked nearby to simplify the human assistant's job of refilling the block pickup zone. The seed block is powered by a 8.4V NiMH battery to provide enough current for sending I2C signals through the blocks to reach the intelligent scaffolding block that would be placed last. The robot would build a horizontal L block structure, already

specified in the construction interpreter script with the command list uploaded to the seed block's memory. The success of the test would be based on how many blocks were successfully placed in the structure and at how many points in the demo the robot would require assistance to overcome any obstacles such as height differences between blocks that the caster wheels could get stuck on.

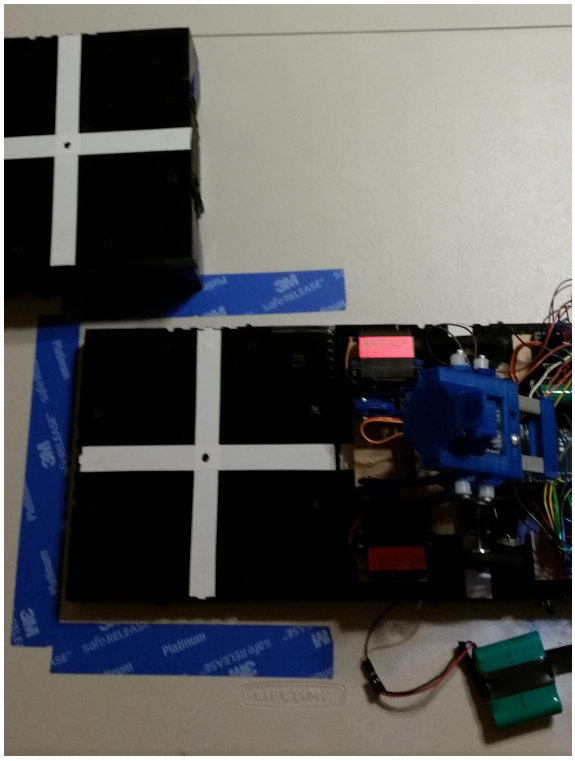


Figure 4.10: Full demo setup, with block pickup zone outlined in blue tape and robot starting on top of the seed block facing the pickup zone (North)

4.6.3 Results

We performed at least 10 full demo tests, but we will mention the results of our first 10 trials here. The first step of the trials, the IR command sequence request, functioned as expected 10/10 trials. Thus, as shown in our unit tests, IR communication is a reliable method for local communication between intelligent scaffolding blocks and the worker robots.

However, due to inconsistencies in the manufacturing quality of blocks—and occasionally the catching of Pogo pins on the copper plates causing a block to be slightly elevated—each trial had at least one or two human assists required. The main human assist required would be when traversing from one build material block to the lower-quality build material block (blue-cornered one) in the structure as it would always have a slight height difference when placed, potentially caused by Pogo pins catching instead of compressing. However, aside from the occasional slight assist required for traversing that block, the robot would always be able to reach its target location in each of the 10 trials. Once the block the robot carried would be placed in position, the robot would then generate the home command sequence, which succeeded in returning the robot to the seed block for further instructions 10/10 times and never required any human assists.

In three of the trials, blocks had to be manually readjusted after placement due to the block spinning into an impossible to correct, roughly 45° offset, which prevents the magnets from adjusting. In another six tests, the block being placed would catch on the corner of the previous block. However, only three of these cases required human adjustment afterwards, as the robot would complete its 180° turn to return home, which would knock the block back into place due to the corrective corner magnets, proving the magnets' effectiveness in a full construction demo. Any other cases of the blocks catching were solved during the robot's end effector raising its linear actuator to separate the magnets, which would cause the block (only slightly caught on a corner) to fall off of the corner into its correct intended position. Thus, despite the manipulation tests showing effectively 100% accuracy, in practice offsets such as not driving far enough onto the center of a block, or having rotational error when overshooting reaching the center of the block cause the robot to place the block ineffectively. Given much stricter manufacturing tolerances for

the assembly of the blocks, as well as more rigid drive motor mounts to ensure the wheels properly grip the acrylic surfaces of the blocks, it is predicted that these manipulation errors would occur much less often.

The robot also never managed to place the leaf intelligent scaffolding block, due to an error with the block sliding off of the magnetic end effector of the robot during placement. We believe that this error could potentially be due to the RGB LED mount in between the central magnet and the underside of the acrylic surface potentially decreasing the magnetic force by serving as a slight obstacle. We considered that the slight increase in mass from the added electrical components such as the Arduino Nano and the IR LED and receiver circuits could have caused the slippage, however we tested this theory by reducing the mass of the block by removing some of the extra corner magnets, and the block still slipped. Thus, with some adjustments to the design of the intelligent scaffolding block to prevent it sliding off of the robot's manipulator during block placement, the reliability of placing the leaf blocks would increase.

Since the robot could not successfully place the leaf block on its own, potentially due to manufacturing errors in the leaf block, we adjusted the failed placement of the leaf block and it connected with the seed block through I2C as expected. When one of the build material blocks would be removed from the structure, the seed block would stop flashing a sequence of colors in sync with the leaf block, and instead the RGB LED on the seed block would shine a constant red to indicate that the leaf block can no longer be found and has been removed from the system. Thus, we show that the ability to detect the status of the structure works in the construction demo.

Chapter 5

Conclusion

5.1 Summary of Background

Construction is a dangerous, but necessary, endeavor that produces the infrastructure we require to live our daily lives. However, every year, hundreds of casualties of construction workers occur at construction sites [4], and thousands more are injured [3]. A potential effective solution is robotic construction, as robots can safely work in harsher environments than humans, and can be easily replaced. Robotic construction solutions also present opportunities to improve existing dangerous build sites such as for underground or underwater infrastructure, and expand construction capabilities to new frontiers for projects such as planetary colonization.

Many robotic construction solutions, especially collective robotic construction (CRC) solutions, take inspiration from nature, such as termites, as shown in the SRoCS [1] and TERMES [26] papers. Termites organize their construction efforts using stigermgy, a process in which markers are left in the environment that provide information to the agents in the system that guides the construction effort. In the termites' case, the termite queen emits pheromones that signals the worker

termites to begin building, and the termite workers lay pheromone trails to guide the other termites to the locations in which they should place their building material [6]. The SRoCS system uses stigmergic blocks to replicate this process, by having the blocks communicate with the robot to indicate if they are seed blocks that request pattern-based structures be built around them, or build material blocks that have already reached the required position in the structure and thus should not be manipulated. Termite nests are also navigable structures built by the termites, ensuring their ability to reach all locations within the structure; The TERMES paper replicates this method by using a system of custom build material blocks that the mobile manipulator robots navigate on top of to place more blocks to complete the structure.

This thesis combines aspects of the SRoCS and TERMES systems to develop a new collective robot construction method. We propose a network of intelligent scaffolding blocks, which serve the same construction-guiding purpose as the stigmergic blocks from the SRoCS system, which allow the robot to navigate along the structure it builds, just like the TERMES system does. By implementing a navigable structure, we can guarantee successful block placement, as the robot's navigation becomes greatly simplified into a physical implementation of a grid world in which all locations in the construction plan can be accessed using a simple set of discrete movements. By simplifying the navigation process, we can easily develop a construction plan interpreter in which a user specifies a 2D structure for the system to build, and the interpreter translates it into a sequence of the basic commands required for the robot to follow to successfully complete the structure. The sequence of basic commands can then be sent to the robot via the intelligent scaffolding blocks, such that the entire construction plan is offloaded onto the blocks and only the specific step of the plan that the robot must complete is stored in its memory, one step

in the plan at a time. By offloading most of the computation that the robot must make, we simplify its design and increase its scalability. Our system thus presents the opportunity to organize multiple robots in a small area using a single intelligent scaffolding block to guide them to ensure smooth construction progress. The nature of the intelligent scaffolding blocks also allows them to be removed from the structure at the end of the building process to reuse them at the next construction site. However, the intelligent scaffolding blocks can also sense the local blocks around them, enabling such benefits as detecting when a block is removed from the structure, which could potentially be applied to helping keep robotic construction systems in the loop of the complete lifecycle of structures (construction, maintenance, demolition) by detecting and automatically performing repairs.

In this thesis, we produced a minimum viable prototype to have a physical implementation of our method that constructs 2D structures. This prototype improves upon the work of Cormier *et al.* [8]. The intelligent scaffolding blocks and build material blocks have a modular design, such that they can be easily replicated and manufactured in batches due to interchangeable parts. Each block has electrical connections running through them, with one side acting as an input side (marked with Pogo pins) and three sides acting as output sides (using copper plates as electrical contacts). These electrical connections allow us to provide a power supply to only the seed intelligent scaffolding block, and have it transfer power along all blocks to reach the leaf intelligent scaffolding blocks that have no self-sufficient power source. However, they also serve a method for the intelligent scaffolding blocks to sense the state of the structure, because once a leaf block is placed, it connects to the seed block through I2C, and if any block is removed between the circuit of the seed block and the leaf block, the seed block can tell that the circuit has been broken.

The intelligent scaffolding blocks use IR communication to send a signal at 38 kHz

to the robot containing a sequence of basic commands, represented by a string of integers, that the robot requests by sending a unique request bit over IR to the block. Once the robot receives the commands, it can construct the structure, one block at a time, returning to the seed block to pick up the next block from the refill depot and to request the next command sequence for the next block in the construction plan. By simplifying the construction process down to a set of discrete commands using our navigable structure framework, we implemented a construction interpreter that translates the 2D structure into a series of commands for the seed block to send to the robot, one at a time. Using a similar algorithm to the construction interpreter, we implemented a method on the robot that generates the path to return from its current location to the seed block, thus shortening the length of the command lists that the intelligent scaffolding blocks must send.

To manipulate the intelligent scaffolding blocks and the build material blocks, the robot uses a permanent magnet end-effector attached to a four-bar linkage which attaches to a permanent magnet in the center of each block. Each block also has magnets in each corner, to allow the blocks to auto-correct their orientation during block placement and to ensure the electrical connections between the blocks line up. This system of magnets increases the robustness of the manipulation, as the robot has a wide margin of error for positioning when picking and placing blocks. Each block has a support magnet as well next to the main magnet in the center, that connects to a support magnet on the robot's magnetic end effector, that prevents the blocks from spinning into severely incorrect orientations during transport.

5.2 Results

We conducted unit tests to determine the reliability of each subsystem of the project before implementing a final full construction demo that uses all of the subsystems combined at once. We divided these unit tests into five categories: manipulation tests, navigation tests, communication tests, construction algorithm tests, and inter-block communication tests.

Manipulation Tests

The manipulation tests were performed to check the reliability of the four-bar linkage and its permanent magnet end-effector in picking up and placing a block successfully. The results of this test would help in diagnosing the cause of any failure points during the full demo.

The robot was set up on top of a block, and tasked with picking up and putting down a block 50 times in a row. The robot successfully completed the manipulation task 48/50 times (the two failures were due to the battery running out during testing). Thus, aside from battery failures, the manipulation is reliable 100% of the time, as long as the block is manufactured properly. 10 pick and place trials were run using a block of lower-quality manufacture, and the robot could only place that block successfully 5/10 times, showing the importance of the quality of the materials. These failures were mainly attributed to the block spinning during manipulation due to a weaker magnetic force from the support magnet, so once the support magnets were doubled, the robot could manipulate even the lower-quality block successfully 10/10 times.

Navigation Tests

We tested both the robot's ability to turn 90° and the robot's ability to navigate between blocks successfully. The robot's original motors were of poor quality and did not meet the torque and speed requirements of our system, thus preventing effective navigation methods from being realized. Once we improved, we conducted these tests to show that our improved navigation algorithms would work successfully, as well as to diagnose main potential source for error in the full demo.

In the 90° turn tests, we placed the robot on top of a block, and tasked it to complete two 90° left turns followed by two 90° right turns on loop, such that the trials were continuous and reflective of potential position error accumulation that could occur when performing multiple turns in the full demo. We determined a success as a turn within only a few degrees of the target of 90°, easily determined by the RGB status LED on the robot chassis. We tested this trial with the robot both loaded with a block, and unloaded (without block). It was found that during the unloaded trial, the robot had success with mainly the second left turn and first right turn, at 18/20 trial accuracy in which the two failures were attributed to high enough accumulated translational error such that the robot could no longer turn on the same axis as the center of the block. The first left turn performed successfully 16/20 times (in which, again, two of the trials were failed due to the accumulated error). The second right turn was the source of the errors, as (potentially) due to light sensor reading differences the turn would almost always overturn and result in a translational offset that could only be partially corrected by the following left turns. The second right turn only succeeded 5/20 trials, and shows the threat that error accumulation poses to the navigation of the robot.

During the loaded turning trials, the robot had much more promising results, as apparently the additional weight of the block assisted in accurate turning. The

left turns and first right turn all succeeded with 20/20, 17/20, and 20/20 successful turns. The source of failure in the unloaded tests remained the source of failure in the loaded tests, but with a much higher success rate of 14/20, such that the error never accumulated enough such that the robot stayed close enough to the center of the block and could complete its turns effectively.

The inter-block navigation tests presented the opportunity to show a different quality of a subsystem, the ability to correct the accumulating errors of a different subsystem. Much like how the magnets correct rotational error during manipulation, the line-following navigation between blocks automatically corrects the translational error that could accumulate during turning in a full construction demo. Thus, we set up two tests, one in which the robot would travel between two blocks (unit test, shortest inter-block travel) and between five blocks (longest straight inter-block travel available to us). The robot would drive across the straight path of blocks until it reached the end block, at which point it would perform a 180° left turn and travel across the blocks again—trials are counted as the path from the first block to the last block and the 180° turn, then the robot's next movement is counted as a new trial. Success was determined by whether the robot traversed the block without getting stuck or driven off-course, and whether the robot's 180° turn was a few degrees within the desired setpoint of 180°.

The inter-block navigation unit test between two blocks had a success rate of 20/20 for movement between blocks, and a turn success rate of 15/20. These trial results show that despite the turns being off by more than a few degrees of the set-point in 5 trials, the robot could successfully navigate between the blocks regardless. 4/5 of those trials resulted in the robot's error being corrected such that the turn following the inter-block travel from a failed turn was successful. The test between five blocks had a similar success rate, with the exception of the robot getting stuck

for a few seconds on the lower-quality build material block due to manufacturing inconsistencies; the robot would ultimately reach the next block though and overcome the slight height difference between the lower-quality block and its destination block. Thus, these results show that the robot can reliably travel between blocks during construction, at least when unloaded. The robot navigating between blocks while loaded with a block was grouped in with the full construction demo testing because of the evident reliability of the unloaded trials.

Communication Tests

We tested the IR LED and IR receiver communication between the block and the robot to ensure that it would work as a reliable replacement for the RGB color commands communication method. We began with a unit test of two Arduino Unos communicating with each other using the sensors, and then concluded with trials using the robot and the intelligent scaffolding blocks.

We set up two Arduino Unos across from each other, one with an IR LED circuit and one with the IR receiver circuit (using a TSOP38328), and tested to see if they could successfully transfer integers increasing from 1-100 (representative of our command sequences). The receiver was moved into various orientations with respect to the IR LED and obstacles were even placed between them, and we discovered that unless a black-surfaced obstacle was placed in such a way that it absorbed the full IR LED's signal, the signal could be received with great success despite obstacles being in the way of the sensors' line of sight to each other. Thus we deemed that the data was successfully transmitted and received so we moved on to implementing the IR systems onto the intelligent scaffolding blocks and robot, and considered the robustness of the IR method to have been proven.

We began by placing the IR receiver and LED underneath the scaffolding block,

however this was quickly realised to be unreliable, as it required the signal successfully bouncing off of the surface beneath the block into the robot's receiver; there was also no way the robot could reliably bounce a signal under the block to communicate with the block. Thus, the intelligent scaffolding block's IR system was mounted above the block's surface to have a direct line of sight to the robot's IR system. The new mounting method was tested using a call-response between the robot and the intelligent scaffolding block. The robot would send a unique request bit to the block and the block would then transmit the command sequence the robot would require to execute a step in the construction plan. Using the robot's RGB LED, the received commands were "executed" through color sequence. The robot indicated it successfully received the full command sequence by displaying the correct sequence of colors after sending its request signal.

Construction Algorithm Tests

We derived the construction algorithm by handwriting test cases to generate the basic commands required to realize the specified structure. Once the rules were generated, we implemented a Python script to translate a CSV representation of the desired structure into a string of commands (represented by unique 1-digit integers). The output of this Python script was compared against the expected results provided by the handwritten test cases.

The output of the construction interpreter Python script matched the expected results from the test cases. When tested against new structures, we solved any edge cases that might occur in the implementation and then confirmed the algorithm's effectiveness by testing even more block configurations. The other configurations tested were also found to be successful, generating the expected list of basic commands required to build the structure. Thus, the interpreter was deemed reliable

for use in our full construction demo.

Using the algorithm for creating a construction plan out of a user-specified structure, we derived the method for reversing the forward command sequence required by the robot to reach its destination into the reverse command sequence required to return the robot home to the seed block to receive further instructions. This algorithm was tested using serial print statements on an Arduino Uno, to increase the speed of testing, and the output generated by the home algorithm was as expected when checked against our derived test cases. Thus, the method for the robot to return home after executing a command was also deemed reliable enough for use in our full construction demo.

Block Communication Tests

The I2C communications between blocks functioned as expected in Enyedy *et al.* [10], thus testing of block communication was completed to ensure that they still functioned properly despite the wear and tear caused by countless manipulation tests. Testing the I2C communications between the blocks would show the reliability of our intelligent scaffolding block system for sensing local areas of the structure, such as a block's removal from the structure that would break the circuit between the intelligent scaffolding blocks.

The intelligent scaffolding blocks were assigned as a seed or a leaf block, and then connected to each other with multiple configurations of build material blocks in between to test if the signal and power could travel across at least five blocks. Initially, the I2C test was performed by powering the seed block with a USB, however it was discovered that it would be more reliable to use a 8.4V battery to ensure that the signal could transfer successfully across all the wires. Once any wiring errors were fixed (due to damage to blocks during transport), the I2C communications success-

fully allowed the seed block to sense the addition of the leaf block to the structure between any configuration of our build material blocks. After sensing the addition of the leaf block, the seed block and leaf block began displaying synchronized color sequences due to the messages the seed block was sending over I2C, showing the reliability of our I2C system for use in the full construction demo test.

Full Demo Test

The full demo tests showed promising results for our implementation of a stigmergic construction system building a navigable structure. The communication between the intelligent scaffolding blocks and the robot functioned properly in 10/10 trials, showing that the IR communication system we implemented is a reliable method for commanding the robot. The navigation algorithms also had successful results, with the robot reaching the destinations specified by the seed block in every test, with the exception of the caster wheels getting stuck on an elevation difference between one of the blocks that had lower manufacturing tolerances occasionally, which the robot would either be able to eventually overcome on its own or would require human assistance in pressing down on the higher block to level the height difference. However, due to the robot's position on some of the blocks during construction or due to a block spinning during transport in a 45° rotational offset, blocks would often be caught on the edges of the previous block during block placement (9/10 total tests experienced this error). 3/9 of these full demo trials did not require the block catching on a corner to be resolved with human assistance, however, as the correction magnets on the corner of each block would readjust the block into the right position once the robot knocked the block off of the corner during its turning procedure to return to the seed block. Another manipulation error throughout the trials was in the robot never being able to successfully place the leaf intelligent

scaffolding block due to it slipping off of the robot's end-effector during placement. The I2C system functioned with similar success as the I2C unit tests, with the blocks being able to communicate with each other once the leaf block was adjusted into place at the end of the demo.

Thus, it is determined that by refining the quality of manufacturing of both the robot and blocks, to ensure consistent hardware properties such as block height, the reliability of manipulation during a full demo could be increased. Consistent hardware would allow for the navigation algorithms to position the robot in correct block placement positions that would not catch on the edge of other blocks, and no blocks would slip or rotate during transport or placement.

Ultimately, the full demo results of our minimum viable prototype show the potential benefits of combining navigable structures with stigmergic construction methods. A system of intelligent scaffolding successfully transmitted a list of basic movement and manipulation instructions for the robot to follow the structure's build plan one step at a time, and the robot executed those instructions and could successfully build the structure with minimal external assistance. The structure's leaf intelligent scaffolding block, once added, allowed for the intelligent scaffolding blocks to detect the state of the system by noticing the break in circuit between the seed and leaf blocks. With an improved physical implementation, this method of collective robot construction could provide another method for building structures at hazardous construction sites.

5.3 Lessons Learned

I learned many lessons throughout my work on this project, and will list the most important ones here.

My first lesson that I learned was that when working with a custom physical robot, never assume that the hardware is adequate until it has been proven reliable. I had initially assumed that the robot from my work on [10] had adequate hardware and the navigation algorithms were preventing the robot from accurately turning and navigating between blocks. However, after many weeks of testing, I came to the conclusion that despite my best efforts programming a successful navigation algorithm for the robot, it would never be able to carry out the intended construction plan due to the drive motors' and wheels' inadequate specifications. Once I calculated the required torque and estimated an appropriate speed for the robot to travel at for successful line following, I upgraded the drive motors and wheels on the robot and was able to successfully implement navigation in a short amount of time soon afterwards. For my future projects I will ensure that I check the hardware, if possible, as well when my algorithm testing consistently has poor execution.

I also learned about the importance of indicators that allow the user to visualise what the state of the system is. When testing the robot's state machine for executing a pre-programmed construction demo, I had some loop errors that I could not diagnose due to being unable to serially connect to the robot while it operated on external battery. Thus, I added an RGB LED just like the ones on the intelligent scaffolding blocks to one of the breadboards on the robot's chassis such that it was easily visible from most angles. The RGB LED would act as my serial print statements during robot demo operation, and it has proven incredibly effective in helping me diagnose any errors in my loops or my state machines during demo testing. Even during the full demo I keep the RGB LED's state indication colors running, to allow the viewers of the demo to understand the changes in the robot's states.

I learned more about stigmergic construction methods and navigable structures as well. By using stigmergy, one can offload construction plans from the robot

to external structures such as stigmergic blocks or intelligent scaffolding blocks, which provides opportunities for construction styles such as building patterns by assembling building material on top of a seed block at the seed block's request (as in SRoCS [1]). The ability to command robots using local communication from the structure itself, as in our system of intelligent scaffolding-based construction also provides interesting opportunities for extension by organizing multiple robots and the opportunity for self-repairing structures by having the intelligent scaffolding sense local blocks in its area and whether the circuit between them and the seed is broken. The navigable structure methods used in TERMES [26] and our intelligent scaffolding system gave me insight into the benefit of simplifying a the navigation required to traverse a construction site by providing a custom structure to travel on. The concepts of collective robot construction that I have learned about throughout my research and design on this thesis have stimulated my interest further into the subject matter—I would like to see these construction systems (or perhaps this one we designed) actively working in construction sites in the near future.

Perhaps most importantly though, I learned the importance of receiving feedback as often as possible from anyone who has time to provide it. The improved motors were recommended to me based on feedback discussing potential causes for the consistent navigation failures. The IR communication method was recommended to me by a colleague who saw me testing the RGB communication and mentioned that IR communication would provide similar capabilities with much greater accuracy (which led me to realize the high effectiveness of IR for simple wireless communication). The lookup table method for performing line following-based navigation as opposed to a full PID controller was recommended by my advisor, and it has enabled the successful navigation of the robot with minimal overhead. Seeking and listening to feedback has allowed for the successful realization of this project, and I

thank everyone who offered me such useful and important ideas for improving my implementation of the minimum viable product of this thesis.

5.4 Future Work

5.4.1 Self-repairing structure

Due to the implementation of the intelligent scaffolding blocks sending commands to the robots, interrupts can be set up to notify the robot that a part is missing from the structure and that it must repair the break before continuing building the rest of the structure that was planned. The scaffolding blocks store the entire plan for the structure, as well as which steps of constructing the structure have already been completed. Thus, if an interrupt caused by the circuit between two intelligent scaffolding blocks is broken, the seed block could issue a repair command to the robot. Once the robot completes the repair task, it could pick up where it left off, as the scaffolding block system remembers the construction progress.

5.4.2 Multiple robots

The intelligent scaffolding construction system provides an opportunity for organizing groups of multiple manipulator robots. Robots could be organized based on the number of seed blocks and block pickup depots available, which would influence the modifications to the algorithm. The robots would also require additional obstacle-avoidance sensors, such as proximity sensors to avoid crashing into their coworkers.

If using only one seed block with one block depot, then command sets could be grouped based on the different branches of construction that must be completed.

For example, if the structure branches off east, west, and south of the block depot, then a command set for each direction could be segmented out in the interpreter, and each direction could be handled by its own robot. Robots could avoid collisions at the seed block by starting their plans at different times and by waiting in queue by detecting other robots using the proximity sensors. The command request each robot would send would have a signature identifying the robot that sent it, allowing the intelligent scaffolding block to determine which command list it should send a command from. In the event of implementing self-repairing structures, the robot that completes the repair interrupt could be the robot that is in charge of the branch in which the break was detected, thus if the interrupt occurs while a robot of a different branch is on the seed block, it can receive its normal building command and the interrupt command could be sent to the intended robot once it arrives and requests its next command.

If using multiple seed blocks with multiple block depots, then command sets could be grouped based on entire subsections of the structure, segmented by modifications to the interpreter. Robots could be placed at each seed block and build the plans for the local neighborhoods of blocks required for their subsections of the construction plan, never trespassing into the other seed blocks' territories, as can be marked using borders of intelligent scaffolding, which can later be removed if placed in more central parts of the structure.

5.4.3 Block placement confirmation

Since all of our intelligent scaffolding blocks can send signals across the wires in the build material blocks, the placement of an intelligent scaffolding block can be confirmed. However, in the current implementation, build material blocks cannot have their placement confirmed as they emit no signals of their own, unlike the I2C

communications between intelligent scaffolding blocks. Thus, if a simple indicator in the build material blocks was also implemented such that the intelligent scaffolding blocks could sense a signal indicating the addition of a build material block, feedback on the success or failure of the robot to complete its task could be used to ensure the proper assembly of the structure even in the event of block misalignment. For example, if the block the robot picks up is supposed to be placed at the location specified by the command set sent by the seed block's communication, the seed block could not advance to sending the next command in the construction plan until after the block's placement is confirmed, such that if the block is placed wrong, a human assistant can remove it from the system and the block will attempt to place it again, as instructed by the intelligent scaffolding block.

Bibliography

[1] Allwright, M., Bhalla, N., El-faham, H., Antoun, A., Pinciroli, C., and Dorigo, M. Srocs: Leveraging stigmergy on a multi-robot construction platform for unknown environments. In *International Conference on Swarm Intelligence* (2014), Springer, pp. 158–169.

[2] Augugliaro, F., Lupashin, S., Hamer, M., Male, C., Hehn, M., Mueller, M. W., Willmann, J. S., Gramazio, F., Kohler, M., and D'Andrea, R. The flight assembled architecture installation: Cooperative construction with flying machines. *IEEE Control Systems Magazine 34*, 4 (2014), 46–64.

[3] BLS. Employer-reported workplace injuries and illnesses (annual) news release. Tech. Rep. USDL-19-1909, Bureau of Labor Statistics, 2 Massachusetts Avenue, NE Washington, DC 20212-0001, Nov. 2018.

[4] BLS. Fatal occupational injuries by industry and event or exposure, all united states, 2018. Tech. rep., Bureau of Labor Statistics, 2 Massachusetts Avenue, NE Washington, DC 20212-0001, Dec. 2019.

[5] Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence 7*, 1 (2013), 1–41.

[6] Bruinsma, O. H. *An analysis of building behaviour of the termite Macrotermes subhyalinus (Rambur)*. PhD thesis, Bruinsma, 1979.

[7] Campo, A., Nouyan, S., Birattari, M., Gross, R., and Dorigo, M. Enhancing cooperative transport using negotiation of goal direction. Tech. rep., University of Namur, 2006.

[8] Cormier, C., Willgress, C. M., and Beeten, N. Robotic scaffolding.

[9] DAGU Hi-Tech Electronic Co., LTD. *Gearmotor*.

[10] Enyedy, A. J., and Sanchez, F. A. Swarm scaffolding mqp.

[11] Everlight Electronics Co., Ltd. *Technical Data Sheet 5mm Infrared LED, T-1 3/4*, 7 2005. Rev. 3.

[12] Grassé, P.-P. La reconstruction du nid et les coordinations interindividuelles chezbellicositermes natalensis etcubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. *Insectes sociaux 6*, 1 (1959), 41–80.

[13] Grushin, A., and Reggia, J. A. Stigmergic self-assembly of prespecified artificial structures in a constrained and continuous environment. *Integrated Computer-Aided Engineering 13*, 4 (2006), 289–312.

[14] Herbrechtsmeier, S., Witkowski, U., and Rückert, U. Bebot: A modular mobile miniature robot platform supporting hardware reconfiguration and multi-standard communication. In *FIRA RoboWorld Congress* (2009), Springer, pp. 346–356.

[15] Khoshnevis, B., Yuan, X., Zahiri, B., Zhang, J., and Xia, B. Construction by contour crafting using sulfur concrete with planetary applications. *Rapid Prototyping Journal* (2016).

[16] Komendera, E., Reishus, D., and Correll, N. Assembly by intelligent scaffolding. *Department of Computer Science, University of Colorado at Boulder* (2011).

[17] Lindsey, Q., Mellinger, D., and Kumar, V. Construction of cubic structures with quadrotor teams.

[18] Napp, N., and Nagpal, R. Distributed amorphous ramp construction in unstructured environments. *Robotica 32*, 2 (2014), 279–290.

[19] Parker, C. A., Zhang, H., and Kube, C. R. Blind bulldozing: Multiple robot nest construction. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)* (2003), vol. 2, IEEE, pp. 2010–2015.

[20] Petersen, K. H., Napp, N., Stuart-Smith, R., Rus, D., and Kovac, M. A review of collective robotic construction. *Science Robotics 4*, 28 (2019), eaau8479.

[21] Pololu Corporation. *QTR-8A and QTR-8RC Reflectance Sensor Array User's Guide*, 2001-2014.

[22] Pololu Corporation. *20D Metal Gearmotors*, 6 2019. Rev. 1.0.

[23] Pololu Corporation. *Pololu Wheel Dimensions*, 9 2019.

[24] Stuart-Smith, R. Behavioural production: Autonomous swarm-constructed architecture. *Architectural Design 86*, 2 (2016), 54–59.

[25] Vishay Semiconductors. *IR Receiver Modules for Remote Control Systems*, 2 2011. Rev. 1.5.

[26] Werfel, J., Petersen, K., and Nagpal, R. Designing collective behavior in a termite-inspired robot construction team. *Science 343*, 6172 (2014), 754–758.