LEARNING TO COLLABORATE
ROBOTS BUILDING TOGETHER

by
Kathleen Sofia Hajash

B.Arch - California Polytechnic State University, San Luis Obispo (2013)

Submitted to the Department of Architecture and the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Master of Science in Architecture Studies
and
Master of Science in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

June 2018

Signature of Author: _____

Kathleen Hajash
Department of Architecture
Department of Electrical Engineering and Computer Science
May 23, 2018

Certified by: _____

Skylar Tibbits
Assistant Professor of Design Research
Thesis Advisor

Certified by: _____

Patrick Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Advisor

Accepted by: _____

Sheila Kennedy
Professor of Architecture
Chair of the Department Committee on Graduate Students

Accepted by: _____

Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee for Graduate Students

Skylar Tibbits
Assistant Professor of Design Research
Thesis Advisor

Patrick Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Advisor

Terry Knight
Professor of Design and Computation
Thesis Reader

LEARNING TO COLLABORATE
ROBOTS BUILDING TOGETHER

by
Kathleen Sofia Hajash

Submitted to the Department of Architecture and the
Department of Electrical Engineering and Computer Science
on May 23, 2018 in partial fulfillment of the requirements for the degrees of

Master of Science in Architecture Studies and
Master of Science in Electrical Engineering and Computer Science

ABSTRACT

Since robots were first invented, robotic assembly has been an important area of research in both academic institutions and industry settings. The standard industry approach to robotic assembly lines utilizes fixed robotic arms and prioritizes speed and precision over customization. With a recent shift towards mobile multi-robot teams, researchers have developed a variety of approaches ranging from planning with uncertainty to swarm robotics. However, existing approaches to robotic assembly are either too rigid, with a deterministic planning approach, or do not take advantage of the opportunities available with multiple robots. If we are to push the boundaries of robotic assembly, then we need to make collaborative robots that can work together, without human intervention, to plan and build large structures that they could not complete alone. By developing teams of robots that can collaboratively work together to plan and build large structures, we could aid in disaster relief, enable construction in remote locations, and support the health of construction workers in hazardous environments.

In this thesis, I take a first step towards this vision by developing a simple collaborative task wherein agents learn to work together to move rectilinear blocks. I define robotic collaboration as an emergent process that evolves as multiple agents, simulated or physical, learn to work together to achieve a common goal that they could not achieve in isolation. Rather than taking an explicit planning approach, I employ an area of research in artificial intelligence called reinforcement learning, where agents learn an optimal behavior to achieve a specific goal by receiving rewards or penalties for good and bad behavior, respectively. In this thesis, I defined a framework for training the agents and a goal for them to accomplish. I designed, programmed and built two iterations of physical robots. I developed numerous variations of simulation environments for both single and multiple agents, evaluated reinforcement learning algorithms and selected an approach, and established a method for transferring a trained policy to physical robots.

Thesis Advisor: Skylar Tibbits
Title: Assistant Professor of Design Research, Department of Architecture, MIT

Thesis Advisor: Patrick Winston
Title: Ford Professor of Artificial Intelligence and Computer Science, EECS, MIT

ACKNOWLEDGMENTS

Thank you Skylar Tibbits, Patrick Winston and Terry Knight for your critical feedback and inspiring conversations. Each of you played an invaluable role in the development and refinement of this thesis. Thank you Skylar for your insightful guidance, pointed questions, and continued support. Thank you for the advice on research, robots, life, and everything in between. Thank you Patrick for inspiring me through your lectures on A.I., the thought-provoking class discussions in 6xxx, and advice on life and writing. Thank you for inspiring my new perspective in understanding human intelligence, A.I., and the human mind. Thank you Terry for always taking the time to give me honest, in-depth, and critical feedback. Thank you for changing the way I think about computation, making and designing and for being there from the beginning of my time at MIT to the end. This thesis and my experience at MIT would not be the same without each of you.

Thank you to everyone at Self-Assembly Lab–especially Skylar, Jared, Schendy, Björn, Mattis, Christophe, and Riley–for an inspiring research environment, insightful conversations, continued support, as well as plenty of laughs, adventures, and ice cream.

Thank you to everyone in the Design and Computation community–especially Athina, Maroula, Oscar, Paloma, and Shokofeh–for the critical discussions, study sessions, feedback, and coffee breaks.

Thank you Caris, Hosea, Igor, John, and Tailin for your invaluable advice and feedback on reinforcement learning.

Thank you to everyone at Architecture Headquarters, CRON, and of

course Inala for making these past two years run smoothly and always being there to help out.

Thank you MIT Department of Architecture, MIT IDC, and the Harold Horowitz (1951) Student Research Fund for the resources, facilities and financial support that made this all possible.

Thank you Kathy, Nava, Melissa, and Adrian for your continued friendship, inspiring conversations, dinner breaks, Skype calls, and unwaivering support throughout the years.

Thank you David for your tremendous amount of support, inspiration, motivation, and patience over the past two years.

Thank you to my parents, Donna and Andy, and my brother, John for sticking by me throughout my adventures across the country. Thank you for the incredible amount of patience, love, and support you have provided me from day one.

This thesis is dedicated to my family, especially my Busia, Dza dza, Grandma, and Grandpa who will always stay with me and continue to inspire me by their courage, love, compassion, and dedication.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

I envision a future where an architect can design a building and send a 3D model or sketch[1] to a group of mobile robots that self-organize to build directly from that model. As the robots start construction, the architect can observe the process, realize that she wants to make changes to the design, update the model, and resend to the robots. They would immediately switch to the new model and make any updates necessary. The robots could be a diverse group with many different functions and skills between them. Some could lay rebar, while others could pour and print concrete. Others may climb walls to detail hard to reach places. They would have different strengths and have to communicate and negotiate roles within the group. In essence, the robots would work together to build the physical manifestation of the 3D model, responding to changes in environment, updates in the model, or surprises in the material. They would have the intelligence to solve complex problems, the social skills to negotiate and communicate, and the physical abilities to build.

From the first industrial robotic arm to Marvin Minsky's Tentacle Arm, robotic assembly has been a focus of research since the beginning of robotics in early 1960s ("Real-World Machines," 1968; Stone, 2004). In practice today, robots are often used for repetitive or dangerous tasks, such as assembly lines in the automotive industry. The setup for assembly lines is extremely time-consuming, requiring a highly skilled technicians to program and test the robots to achieve the high-precision necessary. By repeating the same task, the return on time invested is through speed and efficiency but is typically lacking in flexibility. In academia, some researchers have taken a distributed approach and built many small, cheap robots that can assemble a large structure faster than one could alone. Others have

[1]
When complete control is desired, the final structure could be determined by a 3D model. In other situations such as rebuilding after a disaster, the goal may be to build many shelters quickly but the specifics are not as important. Here, the robots could work with certain parameters (height, width, and enclosure parameters), the site-specific constraints, and material constraints to build an appropriate shelter.

developed new methods of coordination between multi-robot teams. While these approaches make great strides forward, they are often too rigid, not allowing for variability in the environment or changes in material. Small changes could disrupt the entire process. If we are to push the boundaries of robotic assembly forward, then we need to make collaborative robots that can work together, without human intervention, to plan and build large structures that they could not complete alone. I argue for a flexible, collaborative, and interactive system that can adapt and learn over time.

In order to have a discussion about collaborative robots, it is first important to define collaboration in terms of this thesis. There are certain terms that have many definitions and can mean something different to every person. Minsky (2007) described these terms as ***suitcase words,*** as they embody many different definitions all packed into a single word. Like consciousness and intelligence, I argue that collaboration is another suitcase word.

While collaboration has been a focus of research in many disciplines, such as business, politics, and national security, there is no consensus on a general theory of collaboration. Some researchers have undertaken studies and attempted to define common metrics by which we can evaluate collaboration (Thomson, Perry, & Miller, 2007). Collaboration is an evolving process whereby parties with different viewpoints, backgrounds, or resources come together to achieve a goal, either shared or individual, going beyond the capabilities of a single party. Collaboration, cooperation and coordination are often confused because they are similar forms of interaction, but the cooperation and coordination are static structures that can be contained within collaboration.

I define ***robotic collaboration*** as an emergent process wherein agents learn about each other and their environment through interacting and develop new rules and patterns of behavior enabling them accomplish something that they could not do alone. To take a first step towards this vision, I have developed a group of agents that learn how to collaborate and work together to move rectilinear blocks (Fig. 1). These aspects of collaboration and learning are the key differences from current methods in robotic assembly. By taking a collaborative approach to robotic assembly, I can define a strategy wherein the process will consistently be more robust,

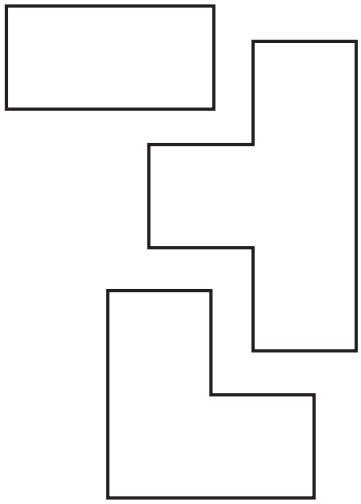


Figure 1.
Three rectilinear blocks that the agents learn to move together.

flexible, and adaptive than coordinated interaction.

To do this, I prioritize the interaction between robots over physical hardware or complex formal designs. Rather than taking an explicit planning approach, I apply an area of research in artificial intelligence called reinforcement learning. Drawn from behavioral psychology, reinforcement learning is a machine learning approach where agents learn an optimal behavior to achieve a specific goal by receiving rewards or penalties for good and bad behavior, respectively. By altering the reward structure to be individual or collectively shared, I evoke different behaviors - for example competitive versus collaborative.

The priority of this research is to develop robots that learn to collaborate to assemble something they could not complete alone. This is the first phase of a larger effort to advance robotic assembly towards utilizing autonomous collaborative robots. More specifically, I propose a multi-robot scalable system wherein robots can operate in dynamic environments and, in real-time, evaluate their surroundings to adjust their next steps.

In this thesis, I define a framework for training the agents[2] and a goal for them to accomplish. Using this framework, I designed, programmed and built two iterations of physical robots. I developed numerous variations of simulation environments for both single and multiple agents, evaluated reinforcement learning algorithms and selected an approach, and established a method for transferring the trained policy to physical robots.

By developing teams of robots that can collaboratively work together to plan and build large structures, we could aid in disaster relief, enable construction in remote locations, and support the health of construction workers in hazardous environments. Collaborative robots would have the ability to evaluate their surroundings, negotiate roles amongst themselves, communicate tasks, and pool their skills. This approach is scalable, robust, and adaptive to changes over time. In addition, this technology would contribute greatly to the architecture, engineering, and construction (AEC) industry in speed and customization.

For example, It does not take much imagination to picture a city recently ravaged by a hurricane. It can take years to rebuild and sometimes months

[2]
In this thesis, agents refers to both simulated agents and physical robotic agents. I explain the terminology between agents, robots and robotic agents in Section 4.1.

to even clear the debris. What if we could deliver a team of mobile robots that could work together to clear debris and rebuild in a particular site? Rather than making one or two huge expensive robots, we could send a dozen smaller robots that could combine their abilities to achieve more than a single robot. This would lead to a more flexible system, where robots would not be limited to their own individual abilities, but those pooled as a group. Other applications may include construction in remote locations such as military deployment, hard to reach sites, or even Mars.

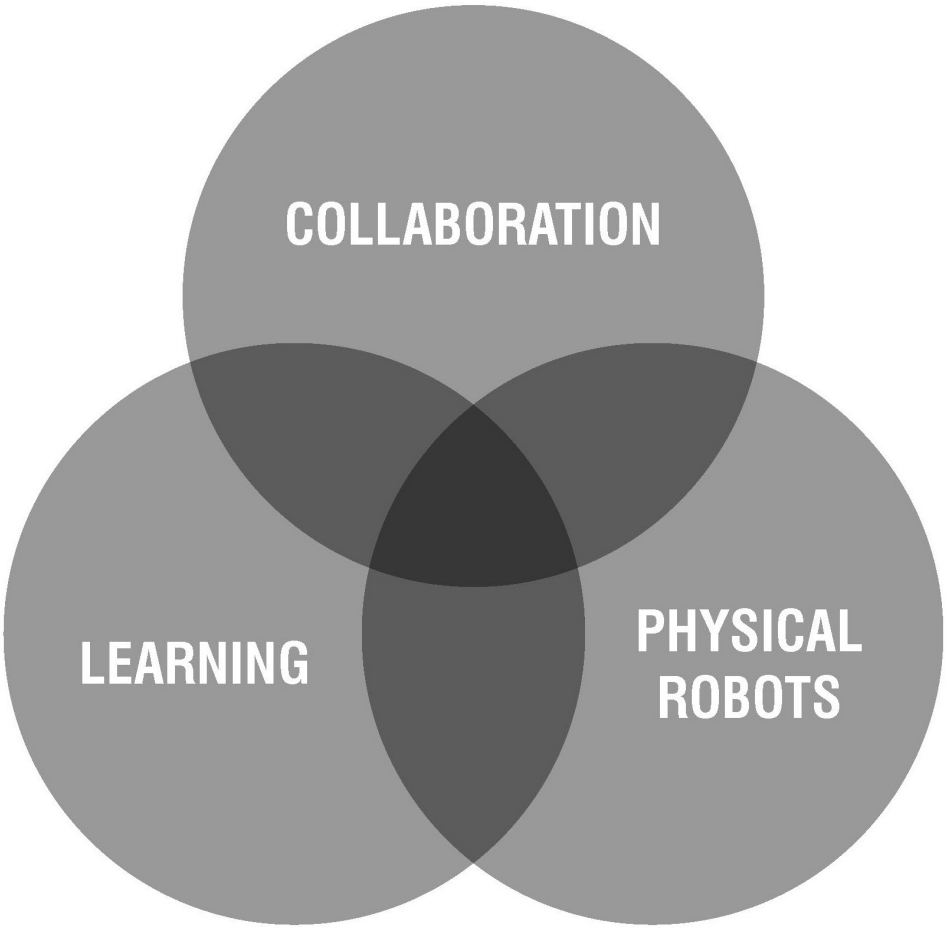
My research is developed in six primary stages: (1) define the toy task; (2) build the robots and the puzzle to assemble; (3) develop the 2D simulation of a single agent and single block; (4) expand simulation to include multiple agents; (5) move to a decentralized approach with each agent learning separately; (6) build framework for learning transfer from 2D simulation to physical robots in multi-agent collaborative environment.

Following this introduction, I cover the following topics. In Chapter 2, I dive into the background information of three important areas of research relevant to this thesis: collaboration, robotic assembly, and reinforcement learning. In Chapter 3, I define a new theory of robotic collaboration, grounded on research presented in Chapter 2.

In Chapter 4, I outline my approach to building collaborative agents. I define a toy task for evoking collaborative behavior. I develop an environment, in both the simulated and real world, for training agents to move blocks. I present my process of testing multiple physics engines and simulation environments, iterating through a variety of environments with the number of agents ranging from zero to ten, and evaluating different reinforcement learning approaches. I build a framework for simulation-to-real-world transfer. Lastly, I review and evaluate my research, highlighting the successes, limitations, and challenges.

In Chapter 5, I present the primary contributions of this research, which include: developing a new theory of robotic collaboration, implementing novel collaborative behavior between agents, developing and fabricating of a team of multiple robots, building a framework for transferring learning from simulation to real world, presenting an evaluation of existing

coordinated interaction between robots, and evaluating reinforcement learning algorithms for methods of collaborative assemblies with multi-robot teams. In addition, I outline future work.


COLLABORATION
LEARNING
PHYSICAL
ROBOTS

CHAPTER 2

BACKGROUND

In this chapter, I outline the three main areas of research relevant to this thesis: collaborative behavior, robotic assembly, and reinforcement learning. In the first section, I give an overview of collaboration from a variety of viewpoints and highlight the differences between collaboration, coordination, and cooperation. In the second section, I present relevant research on robotic assembly and multi-agent environments including robotic assembly lines, agent-based systems, swarm robotics, and multi-robot coordination. Lastly, I present a general outline of reinforcement learning, a subset of machine learning and argue why this approach is relevant.[3]

2.1 COLLABORATION

Over the past few decades, collaboration has received a lot of attention from researchers in fields ranging from microeconomics and business to politics and behavioral science. However, there is still a lack of consensus on a cohesive definition and overarching theory of collaboration (Gray & Wood, 1991; Thomson, Perry, & Miller, 2007). Collaboration can be understood at three different levels: interpersonal or team-based, intra-organizational, and inter-organizational (Colbry, Hurwitz, & Adair, 2014).

The most relevant theory on collaboration comes from Gray who defined collaboration as "a process through which parties who see different aspects of a problem can constructively explore their differences and search for solutions that go beyond their own limited vision of what is possible" (Gray, 1989, p. 5). This description can be used to describe collaboration from the scale of individuals to organizations or even governments. It contains a number of important points worth unpacking. The most important point

Figure 2.
(Opposite) This thesis consists of the intersection of three areas of research: collaboration, robotic assembly, and learning.

[3]
This thesis only includes a general overview of reinforcement learning (RL) for purposes of understanding the research. For readers interested in learning more, I provided detailed resources in the appendix.

is that collaboration is a process–not something that is fixed or static.

Gray expands on this point by describing it as "an emergent process rather than a prescribed state of organization," wherein one can begin to understand the origins and evolution of the organization over time (Gray, 1989, p. 15). In addition, collaboration is an active process where both parties are participating, learning about each other, and searching for a mutually beneficial solution. There is always a phase of exploration, especially at beginning of establishing a collaborative relationship, where each party needs to learn about the other's capabilities, motivations, constraints, and final goals. Through this exploration it is possible to establish a framework for the relationship, or alliance, but it is important to remember that this framework is fluid and will often adjust over time.

Another key point in collaboration is that by coming together, the collaborative alliance can produce a result that is greater than one could alone. This can often be through vision, physical resources, or abilities. Because collaboration is voluntary and both parties can leave at any time, there is typically a mutually beneficial goal that brings groups together, something they could not achieve alone.

COLLABORATION: PRECONDITIONS, PROCESS AND OUTCOMES

In "Collaborative Alliances: Moving From Practice to Theory," Gray and Wood (1991) outline six theories of collaboration by evaluating nine research articles studying collaboration in a variety of settings, including resource dependence theory, social ecology theory, negotiated order theory, etc. The vast range of applications and settings for collaborative alliances makes it difficult for a single theory to emerge. Through their analysis, Gray and Wood (1991) defined three primary components of collaboration–preconditions, process, and outcomes–yet they note that the process is the least studied area.

The ***preconditions*** are the factors that make a collaboration possible. These are the motivations that each party may have and the environmental conditions that facilitated to alliance to form. The players may have a shared vision, or they may have different goals but can only achieve their individual goals by working together.

Next is the ***process*** by which collaboration happens. In their synthesis of theories, Wood and Gray (1991) noted that only three of the six theories addressed process and argue that more research is needed to understand what contributes to the active collaborative process. Those theories that address the collaborative process view it not as a stagnant process or a fixed structure, but as a phenomenon that is dynamic and ever evolving.

The last component of collaboration is the ***outcome***, which includes the metrics by which groups can evaluate the success of the collaboration. To evaluate these outcomes, researchers asked questions regarding which problems were solved and to whom did they benefit, if the alliance transform through the process and did it survive, and if participants agree on shared norms (Gray & Wood, 1991).

Noting the lack of empirical data, Thomson et al. (2007) undertook a research effort to understand and measure the collaborative process. They describe five primary dimensions of collaboration: governance, administration, mutuality, norms and organizational autonomy (Thomson et al., 2007). Governance and administration share similarities in that they are about structural organization. Governance applies to the process of all parties jointly defining the rules and structures that will guide the future of their relationship, while administration is more about management, communication, implementation and coordination. Mutuality refers to the interdependence that both parties share; each benefit from their relationship, but not necessarily in the same way. Norms are mostly about the trust that both parties must develop for one another. As they each contribute to the collaboration, the trust can grow. Lastly, organizational autonomy refers to the fact that each party comes to the table voluntarily and they may have competing interests between their interest of their alliance and the self-interest of their respective organizations.

COLLABORATION, COOPERATION AND COORDINATION

Now that I have established a general understanding of collaboration, it is important to clarify how collaboration is distinct from coordination and cooperation; each is a distinct and separate concept of interaction (Gray & Wood, 1991; Rogers & Whetten, 1982). As Gray and others undertook

an effort to define collaboration in the late 1980s and 1990s, Rogers and Whetten (1982) sought to organize and evaluate the existing research on coordination stemming back to the 1950s. Inter-organizational coordination is a "process whereby two or more organizations create and/or use existing decision rules that have been established to deal collectively with their shared task environment" (Rogers & Whetten, 1982, p. 12).

In clarifying the differences between cooperation and coordination, Rogers and Whetten evaluated the two theories on the following five axes: rules defined, goals emphasized, linkages between parties, resources dedicated, and level of autonomy maintained. In Table 1, I include the five categories as listed in Rogers & Whetten (1982, p. 13) and their descriptions for cooperation and coordination with an additional column for collaboration.

Cooperation (Table 1, Col. B) is the least structured of the three methods of interaction. It requires a significant amount of trust as there are no formal rules for the relationship, but there is also less commitment and risk involved. There are typically no formal linkages between parties, either horizontally or vertically in structure. Each party has their own goals–they may commit fewer resources than coordination and do not abandon any autonomy in their ability to make decisions (Rogers & Whetten, 1982).

Coordination (Table 1, Col. C) is the most structured and formal of the three methods of interaction. It employs specific rules for shared goals and structured linkages between organizations as the relationships can grow complex to avoid inefficiency. As the relationship, linkages and interactions grow in complexity, it may be likely that the organizations have to relinquish some aspects of their full autonomy. In addition, coordination is likely to require high ranking personal in order to make decisions quickly and a significant amount of resources (Rogers & Whetten, 1982).

In ***collaboration***, (Table 1, Col. D) all parties either maintain full autonomy or, in special cases, agree as a group to give certain aspects of their autonomy to the alliance as a whole (Wood & Gray, 1991). In contrast to cooperation, collaboration does mandate rules to guide the process, but all parties come to the table without rules in place and must jointly and explicitly decide what those rules are (Thomson et al., 2007; Wood & Gray, 1991). Groups

[A] Criteria*	[B] Cooperation*	[C] Coordination*	[D] Collaboration
Rules and formality	No formal rules	Formal rules	Jointly define formal or informal rules
Goals and activities emphasized	Individual organization's goals and activities	Joint goals and activities	Both shared and individual goals and interests
Implications for vertical/ horizontal linkages	None, only domain agreements	Vertical or horizontal linkages can be affected	Typically horizontal linkages between organizations
Personal resources involved	Relatively few–lower ranking members	More resources involved – higher ranking members	More resources involved – higher ranking members
Threat to autonomy	Little threat	More threat to autonomy	Maintain autonomy, but can be conflicted between organizational self-interest & collective interest

Table 1.
This table outlines the primary difference between cooperation, coordination and collaboration. The columns labeled with an asterisk (*) are taken from Rogers & Whetten (1982) while the third column is completed by compiling research from Wood & Gray (1989) and Thomson et al.(2007).

can either have shared goals or different goals, but may have complementary resources that make a collaboration mutually beneficial (Thomson et al., 2007; Wood & Gray, 1991).

It is important to keep these definitions of cooperation, coordination, and collaboration in mind throughout the rest of this thesis. The most important axes are the formal structure, level of autonomy maintained, and the goals as shared or individual (Fig 2). To summarize, in cooperation, parties work together to achieve separate goals with few rules guiding their relationship and maintaining their individual autonomy. In coordination, parties work together with formal rules to achieve a common goal, relinquishing a significant amount of autonomy. Lastly, in collaboration, parties work together to achieve individual and shared goals. They jointly decide on rules that guide their interaction and the maintain a significant amount of autonomy. An significant aspect of collaboration is that by coming together, the parties are able to achieve more together than alone. The distinction between the three theories provides a framework for understanding the different approaches to robotic assembly.

In the following section, I describe a number of techniques for multi-robot environments. The majority of researchers describe their multi-robot environments as cooperative or coordinated, while a couple claim

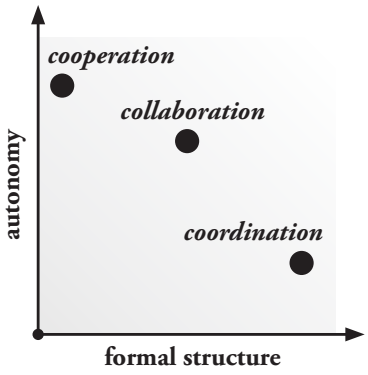


Figure 3.
Diagram mapping cooperation, collaboration and coordination along two axes (autonomy and formal structure). In cooperation, parties work towards individual goals, while in coordination parties work towards shard goals. Collaboration can include both shared and individual goals.

to be collaborative. I present an argument for why none of them are truly collaborative, and why collaboration is vital in advancing autonomous multi-robot assembly.

2.2 ROBOTIC ASSEMBLY

While automated devices powered by water originated in 3000 B.C. with the Egyptians, the first modern robot was not invented until the 1950s (Hall, 1985). This history of robots has deep roots in both academic research and industrial applications. After inventing the first modern-day robot in 1954, George Devol was joined by Joseph Engelberger to develop the first industrial servo-controlled robot, Unimate (Stone, 2004). GM purchased the first Unimate in 1961, and from there robotics began to play a huge role in industry settings taking over tasks that were too dangerous or repetitive, as well as assembly of automobiles, computers, motors, and even robots (Hall, 1985).

Concurrent with industry, research labs were instrumental in advancing robotic research. By 1968, Marvin Minsky had developed the Tentacle Arm which used vision to stack blocks of varying sizes ("Real-World Machines," 1968). Stanford Research Institute (SRI) quickly followed with the Stanford arm in 1969 and developed Shakey in 1972 as the first mobile robot capable of reasoning about and navigating its surroundings (Stone, 2004).

MICRO-WORLDS + BLOCKS WORLD

The idea of working in a simplified world of blocks has strong roots at the Massachusetts Institute of Technology (MIT). In 1970, Minsky and Papert sent out an internal MIT memo describing micro-worlds, a simplified and "fairyland" environment, as a new focus of research arguing that it allows them to focus on important topics without the challenges of the realities in the physical world (Minsky & Papert, 1970). Among the micro-worlds is the "blocks world," used as a robotics environment and a children's story environment (Minsky & Papert, 1971). Patrick Winston (1970) developed a program that could learn to recognize block configurations by being presented examples, near-misses, and non-examples. Separately, Terry Winograd (1971) implemented the blocks world for his program SHRDLU, whereby human users place requests to move blocks using

natural language.

Drawing from this foundation of micro-worlds, I developed a toy problem within my own simulated "blocks world." While this is a simplified problem using blocks, it is intended to simulate a simple physical task in the real world for eventual transfer and not exist solely in the "fairyland" realm. In Chapter 4, I explain my process of defining and building a blocks-based environment. As this thesis is about collaboration between agents, I focus the remainder of this section on approaches that utilize two or more agents, including agent-based modeling, distributed swarm robotics, and multi-robot coordination.

AGENT-BASED MODELING

Agent based modeling (ABM) is a popular approach for simulating complex systems and identifying emergent, bottom-up patterns, especially in political, social, and economic sciences (Bonabeau, 2002). Examples of applications include social interaction, traffic flow (automobile and pedestrians), market simulation, and diffusion, to name a few (Bonabeau, 2002; Chen, 2012; Macal & North, 2005). While there is significant debate about what constitutes an agent, there is agreement that agents are independent, discrete, autonomous, and interact in an environment with other agents (Bonabeau, 2002; Macal & North, 2005). The scale of ABM tends to vary from small simulations capable of handling dozens to hundreds of agents to larger simulations modeling thousands to millions of agents (Macal & North, 2005). Based on this research, I classify ABM as coordinated interaction because agents act based on fixed rules that guide their behavior.

There is some confusion about the differences between ABM and multi-agent systems (MAS). While some recognize ABM and MAS as the same approach (Chen, 2012), Niazi and Hussain (2011) performed a scientometric survey[4] of over a thousand articles covering the concept of "agents," arguing that they are distinct but similar domains. MAS are more associated with artificial intelligence and robotics, while ABM is often applied to the social, biological, and environmental sciences (Niazi & Hussain, 2011).

[4]
Scientometrics is "the quantitative study of scientific communication" (Niazi & Hussain, 2011, p. 4)

While this thesis involves agents and emergent behavior, it is situated more in the realm of MAS than in ABM. As discussed above, ABM focuses on emergent patterns at a scale of interaction (dozens to millions) much larger than the scope of this thesis (two to four). In addition, ABM is more associated with simulating a complex system of interaction between many agents in order to understand more about the system or predict implications of a decision. As a MAS, this thesis employs methods of artificial intelligence to engineer emergent behavior.

DISTRIBUTED SWARM ROBOTICS

With early roots in agent-based systems (Macal & North, 2005), swarm robotics is a decentralized and distributed approach to robotic assembly. Here, researchers utilize swarm robotics to assemble blocks or building components into larger structures. Swarm robotics is "a field of multi-robotics in which large number of robots are coordinated in a distributed and decentralised way" (Navarro & Matia, 2012). This strategy completely eradicates any restrictions on the build volume because the work is split between many autonomous mobile robots.

In the project, TERMES, Peterson, Nagpal, and Werfel (2011) created climbing robots that could stack blocks to build specified structures. As part of an effort to move towards autonomous on-site robotic construction, Melenbrink, Kassabian, Menges, and Werfel (2017) developed a simulation wherein agents could build a cantilevered structure through local awareness, checking the forces of the structure they were building. In typical swarm projects, the robots are all identical and complete the same tasks. They operate autonomously, sensing their local surroundings to avoid obstacles and one another. Unfortunately, the most interaction the robots have with one another is to avoid collision. If a block is larger than expected, these robots cannot work together to move it into place. They must always operate as individuals and can never improvise to lend a hand or arm.

In the distributed swarm approach, the robots are coordinated in that they are working together with a shared goal of building a structure, but they are designed and programmed in such a way that they could operate in isolation or with others. They cannot tell the difference. To facilitate this,

every building component is scaled to the size of an individual robot. If the robots encountered a larger brick or two were accidentally fused together, they would not be able to move it and would require human intervention. In this instance, I argue for robots that can evaluate if they need aid from another robot, call for assistance, and move the larger block together. This, I argue, is robotic collaboration.

MULTI-ROBOT COORDINATION

Over the past decade, architectural researchers have been increasingly exploring fabrication and assembly processes with multiple robots. In 2011, the Southern California Institute of Architecture (SCI-Arc) opened Robot House, a lab with five Staubli robotic arms with intersecting work spheres (Winstanley, 2011). Over the years, SCI-Arc students have demonstrated expertise in planning complex interactions and coordination of multiple robotic arms to build glass sculptures, display light-shows, paint portraits, and more (Testa, 2017). In 2016, ETH Zurich opened their own Robotic Fabrication Lab (RFL) with four industrial six-axis robotic arms (Gramazio Kohler Research, 2016). Unfortunately, while this coordination is impressive, it is also extremely tedious work, requiring careful planning and meticulous, slow test runs.

Other researchers have taken an aerial approach to assembly, using drones to aid in larger scale constructions, unhindered by the limited work-sphere of fixed robotic arms. Augugliaro et al. (2014) used a group of four drones to autonomously assemble a multi-story structure. Felbrich et al. (2017) used drones to bridge the spatial gap between two fixed robotic arms by passing a fiber effector for winding back and forth, similar to a weaving shuttle. Both of these approaches were centrally controlled with a dispatcher sending tasks to the various robots in a predetermined sequence.

Lastly, there has been significant research in assembly with multi-robot teams using a planning approach from artificial intelligence. Here, a planner takes in a variety of constraints and parallelization, returning an optimal plan and set of instructions for completing the task. Dogar et al. (2015) developed a multi-scale perception system that utilizes computer vision in combination with other sensors to alternate between different levels of

precision when assembling a scaled-down plane wing. Knepper, Layton, Romanishin, and Rus (2013) built a series of pre-planners, planners and control systems that can take data on individual components and determine the way the pieces fit together, an assembly sequence and lastly dispatch the instructions to a group of robots.

While Dogar et al. (2015) and Knepper et al. (2013) make great strides in multi-robot assembly, their systems are rigid with specific modes of interaction between the agents. For example, when the robots are transporting a panel in Dogar et al. (2015), they execute an algorithm called "fleet control," where each robot grabs the panel at some location, then the robots move as one to transport the panel. If there is a narrow pathway or obstacles where the robots need to reorganize around the panel, they would not be able to do so.

For all of the projects described above, I argue that the missing ingredient is collaboration. All of the multi-robot teams act in coordination, but not collaboration. Coordination, as described in Section 2.1, is a predetermined process with a fixed set of rules for interaction. In these projects, the robots worked together to execute pre-planned steps without the ability to improvise interaction or change steps based on current environmental factors. Their interaction with each other is specified and does not vary. In contrast, collaboration is a more flexible process wherein robots work together to complete tasks through responding to their environment, improvisation, and defining their own rules for interaction.

2.3 REINFORCEMENT LEARNING

Concluding this review of robotic assembly, I argue that collaborative behavior is the missing ingredient in current approaches. By interacting with each other and their environment, robots can learn how to work together to achieve something that they could not do alone. In order for the agents, simulated or physical, to learn, I employed an approach called Reinforcement Learning (RL). Drawn from behavioral psychology, RL is an area of research within machine learning (ML) and artificial intelligence (AI) whereby agents learn behavior by interacting with their environment and receiving reward signals as feedback.

In this section, I present an overview of reinforcement learning, outline recent advances in deep reinforcement learning (Deep RL), and evaluate relevant research in multi-agent and multi-robot domains, as well as simulation-to-real-world transfer.

ROBOTICS + REINFORCEMENT LEARNING OVERVIEW

Reinforcement Learning (RL) is based on Markov decision processes (MDPs) or, as is often in robotic situations, partially-observable Markov decision processes (POMDPs). In an MDP, an agent exists in an environment in discrete time (Fig. 4). At each timestep, the agent receives an observation about the current state `s` of the environment. The agent then chooses an action `a` to perform in the environment. As a result of that action `a`, the agent transitions to a new state `s'` and receives a reward `r` (Sutton & Barto, 2017). A formal definition of an MDP is a 5-tuple that contains: a finite set of states `S`, a finite set of actions `A`, a transition function that maps the probability that an action `a` in state `s` at timestep `t` will result in transitioning to state `s'` at timestep `t+1`, a reward `r` after each transition, and a discount factor.

Agents are "complete, interactive, goal-seeking" meaning they "have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments" (Sutton & Barto, 2017, p.3). In a POMDP, the agent is not able to see the entire state of the world; therefore, it is partially-observable. Rather than receiving the full state of the environment at each timestep, the agent receives an observation of the environment from its perspective.

With MDPs and POMDPs, the agent has either a probabilistic or deterministic model that maps states to action. In other words, it has a policy that tells the agent what action to take at each timestep. In RL, the goal is to learn the optimal policy that maximizes its reward. Over the years, researchers have developed a variety of approaches to learn the optimal policy, and these primarily break down into two categories: ***dynamic programming*** including value iteration, policy iteration, and Q-learning; and ***policy gradient methods*** including actor-critic policy gradient and monte-carlo policy gradient.[5]

[5] *For more detailed information about these approaches and the foundations of reinforcement learning, (Sutton & Barto, 2017) is a keystone resource.*

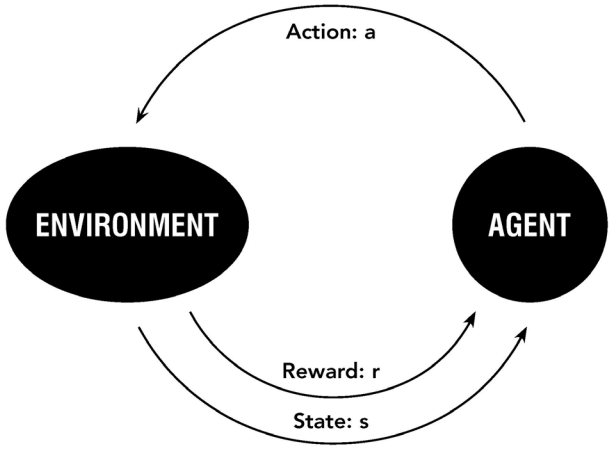


Figure 4.
Diagram showing cyclical agent-environment interaction in an MDP. Adapted from Sutton & Barto (2017, p. 38)

To understand RL, it can help to think about it in terms of training a dog (Fig. 5). In this example, the dog is the agent, and the trainer is the environment. The trainer gives the dog a signal, for example a hand sign to sit. The dog receives this observation from the environment. In reaction, the dog either sits or not. If the dog successfully sits, then the trainer rewards it with a treat, i.e. positive reinforcement. If not, then the trainer does not award the dog anything and continues to send it the signal. In RL, instead of treats, the agent receives numeric points, either positive or negative, as rewards. Every time the dog sits, the environment resets to its initial state with the dog not sitting and no signals. Then training begins again.

This is an example of an episodic task where the agent learns through a series of episodes that end when the agent enters the terminal state or reaches the maximum number of steps in an episode, whichever comes first. A continuing task does not have a terminal state and has an infinite horizon in terms of time. As the aim of RL is to maximize its reward, a continuing task has an additional challenge when it comes to finding a maximum over infinite time. To handle this issue, RL employs the use of a discount rate, a parameter [0, 1], that makes earlier rewards have a higher impact over rewards later on in time (Sutton & Barto, 2017). The task that this thesis addresses is an episodic task; therefore I focus my attention on episodic learning.

Action and state (or observation) spaces can either be continuous or discrete. Using an action space as an example, continuous refers to when the action can be any floating point number within a specified range, such as linear

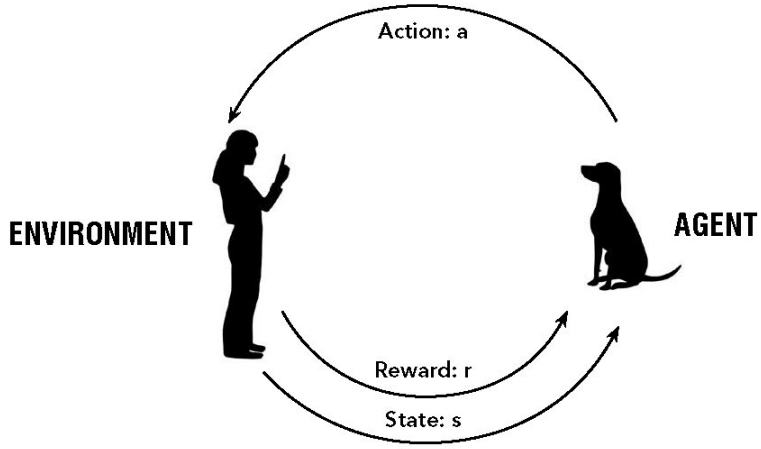


Figure 5.
Reinforcement learning diagram comparing a dog to the agent and a trainer to the environment

velocity. In contrast, a discrete action space consists of a set of separate and distinct actions from which one of those actions can be selected at any time. For example, up, down, left and right could be a discrete action space.

In addition, rewards are often shaped or discrete. The example of teaching a dog to sit =(Fig. 5) is an example of discrete rewards. The dog either receives a +1 or a 0 at the end of each episode based on whether the it successfully sits or not. Shaped rewards are often used in continuous state (or observation) space to give the agents hints to the goal state. For example, if the goal of an agent is to move to a specific target, a shaped reward would be a living penalty based on the agent's distance to the target. If the agent is far away, it receives a larger penalty than when it is close to the target. Video games are ideal for RL because of the preprogrammed rewards for achieving success in the game and the discrete set of actions from which to choose.

In terms of robotics, there are three primary approaches to programming robots and giving them new skills: direct programming, imitation learning, and reinforcement learning (Kormushev, Calinon, & Caldwell, 2013). Direct programming is a low-level approach and tends to be more deterministic. Imitation learning includes multiple approaches, from teleoperation, to physically moving the robot into place and recording the motion, to the robot learning by observing a demonstration of the task (Kormushev et al., 2013). In contrast, the main ambition of robotic RL is to give the robots the "ability to learn, improve, adapt and reproduce tasks with dynamically changing constraints based on exploration and autonomous learning" (Kormushev et al., 2013, p. 122). While there are two primary approaches–

value-function-based and policy-search–in robot reinforcement learning, policy-search is more common because it can handle high-dimensional state and action spaces (Kober, Bagnell, & Peters, 2013).

ADVANCES IN DEEP REINFORCEMENT LEARNING

Over the past five years alone, researchers have made great strides forward against some long-standing challenges in RL. Neural Networks (NN) and Deep Neural Networks (DNN)[6] in combination with hardware advances have enabled researchers to work on harder problems and even work directly with images (raw pixels) as input. Depth of a network refers to the number of hidden layers used in the network, with deep generally meaning more than one. Since the breakthrough in image classification in the annual Imagenet competition (Krizhevsky, Sutskever, & Hinton, 2012), convolutional neural networks (CNN) and deep neural networks have become increasingly popular.

Drawing from recent advances in image classification and speech recognition, researchers at Google DeepMind integrated CNNs with a version of the traditional Q-learning algorithm,[7] presenting a new algorithm Deep Q-Network (DQN) that learned how to play seven Atari 2600 games purely through pixels (Mnih et al., 2013). DQN could take high-dimensional continuous data as input, opening the door to using raw pixels as input without post-processing or specified contour and shape detection. Unfortunately, DQN was restricted to low-dimensional discrete data as output, therefore limiting its applications. It is possible to discretize[7] continuous actions, but for complex controllers for robotic arms with 6-axis, the discretized spaces become intractable–also known as the curse of dimensionality (Bellman, 1957).

To address this issue of continuous control in action spaces, Lillicrap et al. (2015) developed an algorithm called deep deterministic policy gradient (DDPG), based on deterministic policy gradient (Silver et al., 2014) and taking insights from DQN (Mnih et al., 2013). Using the same network architecture and hyper-parameters, DDPG was able to find competitive policies for over 20 simulated physics-based tasks. In addition, they demonstrated the algorithm's ability to learn from low-dimensional

[6] *To learn more about neural networks, convolutional neural networks and deep learning, I have a section on learning resources in the appendix.*

[7] *Discretizing refers to the process of turning a continuous action, such as velocity, into a discrete set of actions.*

observational data (location, velocity, etc.) and high-dimensional data such as raw pixels, also called "end-to-end learning."

In RL, there is a trade-off between exploration and exploitation. Exploitation refers to following the current policy by selecting the best known action, while exploration would be trying a new action (Sutton & Barto, 2017). This can be compared to a common example of selecting a restaurant for dinner. Exploitation would be choosing your favorite restaurant, while exploration would be choosing a new restaurant which could potentially be your new favorite or be considerably worse. A standard approach to adding more exploration is to add noise to the action selection process. In recent advances, it was found that adding parameter space noise to existing RL algorithms can improve exploration and performance (Plappert et al., 2017). This approach was demonstrated using both algorithms mentioned above, DQN (Mnih et al., 2013) and DDPG (Lillicrap et al., 2015), as well as TRPO (Schulman, Levine, Moritz, Jordan, & Abbeel, 2015).[8]

Lastly, there is great interest in the interaction between multiple agents and understanding how they evolve together, both in competitive and cooperative environments. Lowe et al. (2017) expanded on the DDPG algorithm to include learning for multiple agents. They present a general-use and flexible multi-agent learning algorithm called multi-agent deep deterministic policy gradient (MADDPG) (Lowe et al., 2017).

MULTI-ROBOT + MULTI-AGENT RL

To demonstrate the variety of applications MADDPG could be applied to, Lowe et al. (2017) built eight multi-agent environments including cooperative navigation, keep-away, and predator-prey, among others. The environments were either cooperative or competitive, and some included aspects of communication. In earlier work in the multi-agent domain, Mordatch and Abbeel (2017) demonstrated how simulated agents could develop compositional language in order to collaborate and achieve specific goals. They trained the agents using reinforcement learning with carefully designed goals to encourage communication and coordination. These projects is an ideal example of using reinforcement learning to promote group interaction and complex behavior.

[8] *Baseline versions of these algorithms are all posted on OpenAI's github account under baselines. https://github.com/openai/baselines*

Working with robotic reinforcement learning poses many challenges, and this can quickly become intractable when working with multiple mobile robots. Amato et al. (2015) developed a tractable approach to solving decentralized partially observable Markov decision processes (Dec-POMDPs) using macro-actions that extend over time (e.g. go to drop-off area, pick up small box, etc.) rather than selecting actions at each timestep. Then they used memory-bounded dynamic programming to solve the Dec-POMDP with macro-actions, and develop a deterministic policies for the robots to follow in the real world.

In 1997, Matarić built four foraging robots that learned to move small pucks from one place to another. To minimize the learning space, she used a fixed set of four programmed behaviors (safe-wandering, dispersion, resting, and homing) and a state space that consisted of combinations of four conditions (have-puck, at-home, near-intruder, and night-time). The agents learned a value function that mapped conditions to behaviors. Both Amato et al. (2015) and Matarić (1997) minimized the search space by simplifying the state and action spaces into predefined states or packaged actions allowing the algorithm to learn the mapping between the actions and states that are temporally extended.

Researchers at X and Google Brain[9] have developed a process for speeding up the training process by using multiple robots (Yahya, Li, Kalakrishnan, Chebotar, & Levine, 2016). In this experiment, they set up four robotic arms to work separately to open a door. Each door handle was slightly different, creating variability in the environment they were engaging. At specific time intervals, the robots would sync their training data to share what they've learned. The idea is that when they sync their data they are sharing their experiences and progressing much faster than they would alone.

Because training in the real world is costly in both time and resources, simulations can be an another alternative. Unfortunately, there are many challenges in transferring learning from simulation-to-real-world due to the inconsistencies between the two environments. Some approaches include improving simulation or learning more robust control policies, but these often run more slowly in training or lose performance in the real world (Christiano et al., 2016). One recent approach addresses these challenges

[9] *X, formerly Google [X], is Google's "moonshot factory" that is responsible for projects such as Loon and Google Glass. Google Brain started as a deep learning and artificial intelligence project at X, but grew so successful that it became its own research team at Google.*

by starting with a poor inverse dynamics model of the robot and learning to improve the model through training (Christiano et al., 2016). This method works with motion planning, learning, and optimization among other approaches for generating policies through simulation. Bousmalis et al. (2017) addressed the challenge of learning transfer in the context of a robot grasping task by adding layers of randomization (visuals and dynamics) as well as feature and pixel-level domain adaptation. Given a rendered image from the simulation, their GraspGAN model could generate synthetic images that resembled true images of the scene.

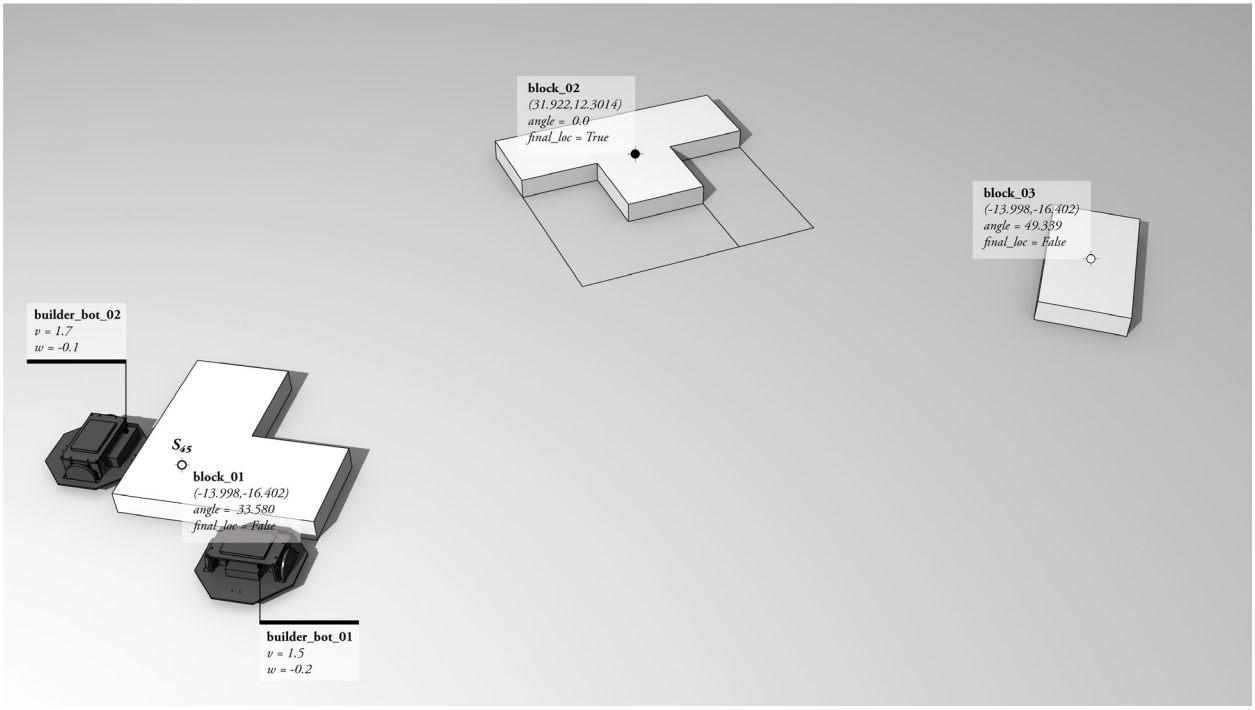


Figure 6. *Goal task: robots collaborate to move blocks to complete puzzle in specified location.*

CHAPTER 3

A NEW THEORY OF ROBOTIC COLLABORATION

Building upon the three stages of collaboration–preconditions, process, and outcomes[10]–I present a new theory of robotic collaboration. Here, I define ***robotic collaboration*** as an emergent process through which individual agents (simulated and physical)–which may have different knowledge, abilities, or intelligence–interact with each other and their environment and learn how to work together to achieve a common goal. Agents with similar or different abilities work together to achieve more than they could alone. They start without any knowledge about the environment or each other and have to interact and explore in order to learn what to do.

In the context of Reinforcement Learning (RL), Sutton and Barto (2017, p.3) define an agent as "complete, interactive, and goal-seeking," and they explain that an agent could be a component of a larger systems, such as a monitor for a robot's battery-level. In this setting of robotic collaboration, I revise this definition as a collaborative agent to be complete, interactive, social, collaborative, and goal-seeking. As Sutton and Barto (2017) encourage a broader concept of what a complete agent could be, I encourage researchers to conceptualize a collaborative agent at multiple scales–as an individual, autonomous robot or as a smaller component of a larger system controlling a robot, building or ecosystem. In the complex robot, certain components, such as the cooling system, vision system and power supply, could be formulated as separate agents that collaborate with each other in order to achieve an shared goal.

Robotic collaboration goes beyond typical robotic assembly because the collaborative and learning-based approach is scalable, robust, and adaptive. First, robotic collaboration is scalable because more robots are not only able to accomplish a task faster but they can also combine their abilities to

[10] *These three stages are outlined in Section 2.1*

achieve more than they could alone–either by pooling the same strengths or combining different skillsets. In contrast, swarm robotics is limited by the capabilities of a single robot where adding more robots only multiplies the capabilities but does not generate more potential. Second, robotic collaboration is robust because is learning is often more flexible in the physical world than more deterministic, planning-based approaches in robotic coordination. In robotic collaboration, there is no prescribed structure for interaction or pre-packaged actions to select. Lastly, robotic collaboration is adaptive because the robots can learn from experience and adjust over time based on the circumstances.

PRECONDITIONS: ENCOURAGING INTERACTION

In the context of robotic collaboration, I define the preconditions as the environment developed by the designer to promote collaboration. The environment includes a task that is difficult to complete without the combined abilities and interaction between the multiple robots. As described earlier, the environment could be the physical world in which robots are assembling a chair or the environment could be the entire control system of a robot.

In one instance, multiple robots may come together to build a shelter after a natural disaster. One robot has the parameters and constraints for designing and building the shelter, another set of robots have more adept vision capabilities, while another set of robots has the strength and dexterity to assemble the structure. The robots have different capabilities, knowledge, and intelligence, and by coming together, they are able to build the shelter. The common goal, differing abilities and physical environment are the preconditions that encourage interaction and collaboration.

The toy task that I have defined in this thesis is for the agents to move a heavy block to a specified location. The block is too heavy for a single robot to move easily, in addition, the robots to not have any way to grasp or latch onto the block, therefore, manipulation is difficult alone. Some of the physical parameters of this environment include: type of control system; strength of agent; vision or observation; size, shape, and density of block; margin of error for goal; number of blocks; number of robots.

Another component that can drastically change the agent's behavior and interaction with others is the reward structure. Typical reinforcement learning experiments use a shared reward for coordination and separate rewards for competition. I argue for a third method of structuring the reward similar to a concept in negotiation called "value creation." This would consist of having separate rewards structures for each agent, but rather than being purely competitive, both agents benefit from actions where they work together.

PROCESS: EMERGING BEHAVIORS

In previous research outlined in Section 2.1, there was less theory and understanding on the collaborative process, more so than the other two phases. This thesis provides an opportunity to develop specific preconditions that will facilitate collaboration and to evaluate the types of collaborative behavioral patterns that emerge. While successful outcomes and solutions are important, these emergent behaviors are the priority of this research. These emergent and sometimes unexpected behaviors are one of the real strengths of this process, in comparison to typical planning or even RL-based approaches that use pre-programmed actions. In those instances, the behavior is always expected and defined in advance, not allowing for much flexibility in application.

OUTCOMES: EVALUATING SUCCESS

While secondary to the emerging behaviors, it was important to determine the metrics by which I can evaluate the success of the collaborative agents. The standard approach in RL is to plot the reward history and number of timesteps necessary for each episode. Depending on the task, an increase in timesteps (e.g. an agent learning to walk) or a decrease in timesteps (e.g. an agent completing a tasks) may be desired. Another approach is to render and record the trained agents for a series of episodes and evaluate the interaction between agents. In this way, I classified the variety of techniques the agents learned for collaborating to move and manipulate the block in my toy task. I evaluated how quickly the agents were able to move the block into place, how frequently they could achieve the specified goal, and how quickly they were able to learn how to work together.

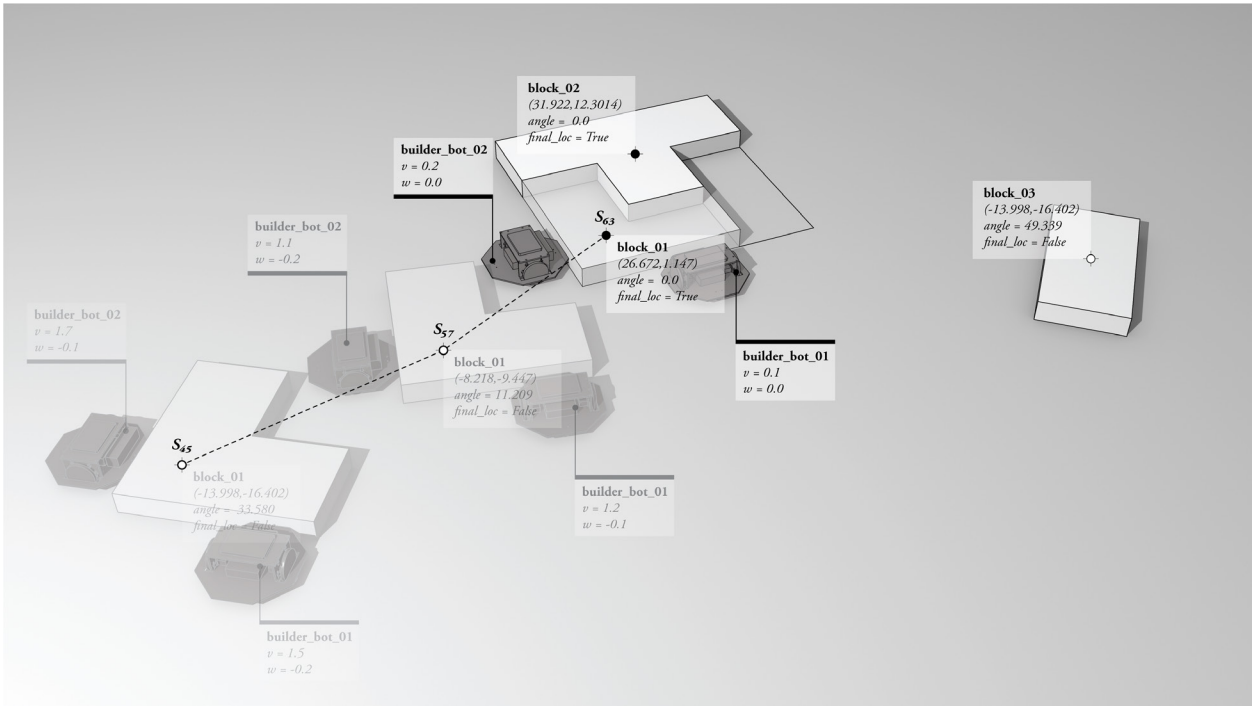


Figure 7. *Conceptual diagram of robots moving blocks into place*

CHAPTER 4

APPROACH

In this thesis, I address the challenge of multi-robot assembly by developing collaborative agents that learn how to work together to move blocks. In Chapter 2, I describe relevant research on collaboration, multi-agent environments, and reinforcement learning. In Chapter 3, I present a new theory on robotic collaboration and argue how this goes beyond current practices. In this chapter, I describe my approach to collaborative robots and discuss how my methods have evolved and shifted over time. I begin by defining a toy task–three blocks for the agents to assemble into a puzzle (Fig. 7)–and end with a framework for transferring learning from simulation to real-world.

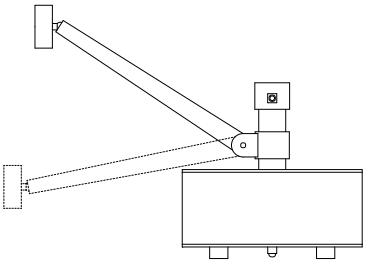
A significant part of my overall process has included repetitively simplifying the problem that I am addressing in order to make it tractable. The problem I have chosen is significant because it includes a multi-agent environment, agents manipulating an object, and a multi-step task. In addition, operating in continuous action and state space means there is essentially an infinite number of state and action pairs possible, meaning it is highly unlikely that it will ever be in the same state again. Other challenges exist around defining a task that is both learnable and general,[11] choosing the description for the state space, and defining the reward function.

4.1 DEFINING THE TOY TASK

From the beginning, I knew it was important to the minimize complexity of the robots and the structure for them to assemble. With this in mind, I sought to create a simple structure–an arch composed of five blocks (Fig. 8). I designed a robot with four dimensions of control; it had a round base, two motorized wheels, and a single axis arm controlled by a servo

[11] *The term "learnable" means that the task is simple enough that the agent will be able to find a solution. While the term "general" means that the task is not too specific that the policy is overfitting and would not be able to find a solution in any alternative setups.*

near the base. An electromagnet was placed at then end of the arm with a universal joint connecting it to the arm. The joint made it difficult for the robot to move a block in a controlled way alone, but it also allowed for more flexibility when transporting a block with other robots. Overall, the four dimensions of control were the two wheels, the single axis arm, and the electromagnet for picking up blocks.



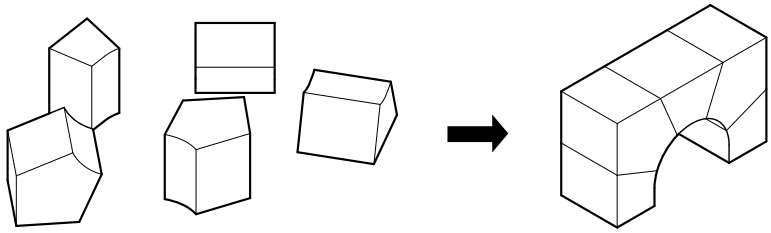
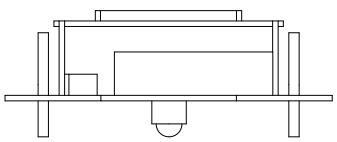


Figure 8.
(Left) Builder Robot 0.0 concept design. (Right) Toy task of building an arch out of five blocks

As I was evaluating next steps, I realized that, even in simulation, multiple agents learning to assemble an arch in three-dimensional continuous space was still too difficult of a challenge to begin with. To simply the problem, I decided to minimize unnecessary dimensions of complexity, in both the robot and environment design, by removing the third dimension and moving to a 2D (or 2.5D in the physical world) environment. Now, the robots cannot lift or grab objects but only push around blocks with their chassis.

Next, I addressed the structure for them to assembly. Moving from 3D to 2D, I could no longer frame the problem as a spatial architectural structure, so I sought to create a puzzle composed of three distinct blocks. The rectilinear blocks, resembling pieces of the game Tetris, were selected from a 9-square grid and fit together to form a square (Fig. 9). Each block is unique in size, form, and mass, with the heavier blocks difficult for one robot to move alone, forcing collaboration.



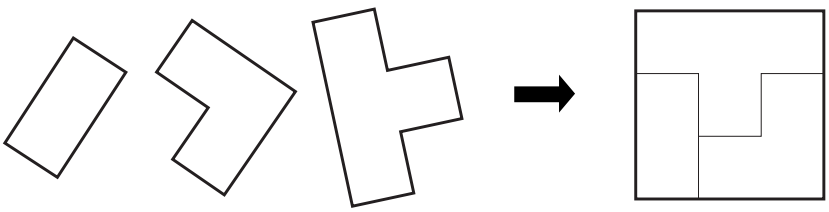


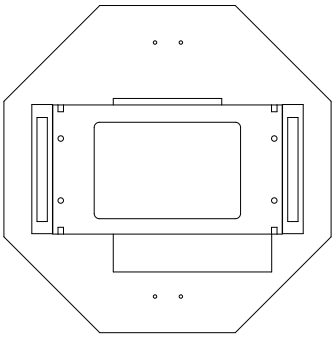
Figure 9.
(Left) Builder Robot 1.0 concept design. (Right) Toy task of assembling a puzzle out of three blocks

Through this design process, I defined a toy task that can be implemented in both a 2D simulation environment and the real-world. I developed the toy task by designing the physical robot and blocks first because my ultimate goal is to implement this in the real-world. In the end, the goal of each agent, simulated and physical, is to move the blocks to their specified final location and rotation. At the beginning of each episode, the blocks are initialized in random locations within the work area, and the agents have to move the blocks to their final location within a certain margin of error.

Before moving on, it is important to take a moment to discuss terminology used in this thesis–specifically the use of agent, robot, and robotic agent. ***Agent*** is the most general term and applies to both the simulated agent and the physical robotic agent. ***Robot*** refers to the physical robot and is typically used when discussing the physical hardware or software of the robot or the vision system for identifying the robot in the real-world implementation. ***Robotic agent*** refers to the physical implementation of the agent acting with a learned policy or in training.

4.2 BUILDING THE ROBOT

As discussed above, it was important to not make the robot any more complicated than necessary to exhibit collaborative behavior. I designed and built the second iteration of the simple robot (Fig. 10) and the lightweight blocks it would assemble. This robot was small, about 6" in diameter, and



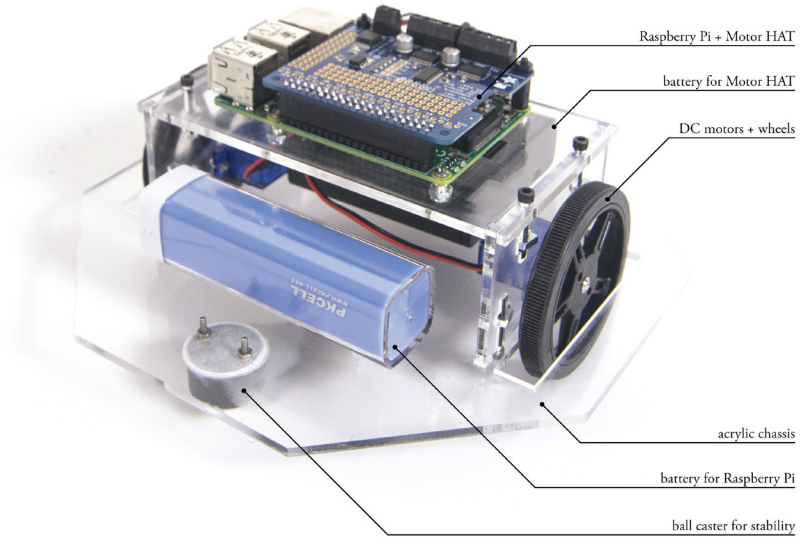


Figure 10.
(Left) Octagonal plan of Builder Robot 1.0 (Right) Builder Robot 1.0

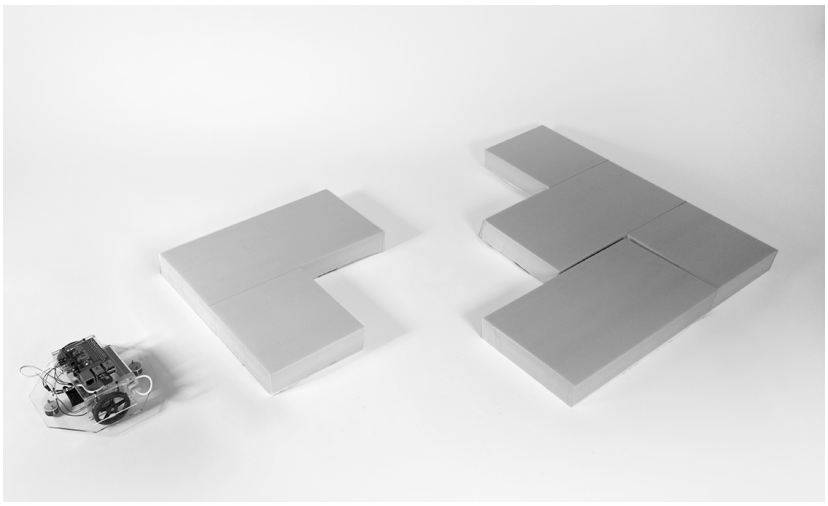


Figure 11.
Manually controlled Builder Robot 1.0 pushes block into place.

it was controlled by a Raspberry Pi 3–a tiny computer the size of a credit card enabled with bluetooth and wifi. A stepper/DC motor hat, sitting on top of the Raspberry Pi, allowed easy control of the DC motors that powered the two wheels.

The motorized wheels were centered on the chassis, allowing the robot to have tight and controlled movement. Two ball casters, located in the front and back, stabilized the robot. The robot did not have any armatures and interacted with its environment solely through pushing around blocks with its chassis. The plan of the chassis was in the form of an octagon–providing the robot with plenty of pushing surfaces for moving the blocks. The edges allowed the robot to push the blocks at different angles, as well as turn the block by rotating its chassis.

The blocks were made from a lightweight extruded polystyrene foam for ease of movement (Fig. 11). An overhead camera provided a global view of the work area. By building the robot first and testing its capabilities, I was able to build a more realistic simulation environment for training.

To control the robot, I developed a custom control interface using TouchOSC, an application that sends and receives Open Sound Control (OSC) or MIDI messages over wifi. A desktop application enables users to design a custom interface with buttons, sliders and toggles. Using the python library, PyOSC, I developed a control system for the robot that listened for instructions from my custom phone application.[12] The controller sends a rotation angle and desired speed to the robot. In Section 4.9, I revisit

[12]
PyOSC is not currently up-kept and only works with Python 2.7, not 3.5+. Python-osc is an alternate library that is up-kept as of the time of this thesis.

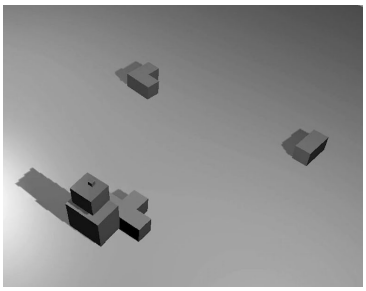


Figure 12.
MORSE manually controlled simulation environment

the robot's physical hardware and software and make the improvements necessary for transferring learning from simulation to the real-world.

4.3 DEVELOPING THE SIMULATION ENVIRONMENT

After building the robot and the blocks it interacts with, I moved on to the simulation and choosing a physics engine. In my first attempt, I designed an environment in Blender, an open source 3D modeling environment and used Modular OpenRobots Simulation Engine (MORSE) for the simulation environment and physics engine. Using linear velocity and turning angle, I manually controlled a robot to test the environment (Fig. 12). While adapting the simulation for Reinforcement Learning (RL), I ran into some key issues in resetting the environment for training. When training an RL agent using episodic time, resetting an environment to its initial state is a mandatory feature.

Because of this keystone issue, I decided to evaluate different physics engines and simulation environments with more documentation and support for RL. After looking at pyBox2D, MuJoCo and Bullet, I decided to start with pyBox2D because of the ease of setup and ability to develop quick proof-of-concepts. MuJoCo and Bullet are both 3D simulation environments that include 3D rendering and a physics engine. They are both popular in the research community in academia and industry, used by groups such as OpenAI, Google Brain, DeepMind, Standford AI Lab, and University of Washington.

Moving forward, I decided to use PyBox2D, a 2D physics engine and simulator, and PyGame for 2D rendering. I developed a manually controlled 2D environment to test out the controls (Fig. 13). At the beginning of each episode, the blocks are randomly placed within the field and the goal is to complete the puzzle as fast as possible, using the white octagonal agent to push the blocks. Small blue dots near the center of the screen mark the final location of each block. The game checks if each block is in its correct location and rotation given a specified margin of error.

In this initial simulation, the actions available are solely the linear velocity in the x and y-axis. These controls are similar to holonomic control but lack control of the angular velocity. Holonomic control refers to when the

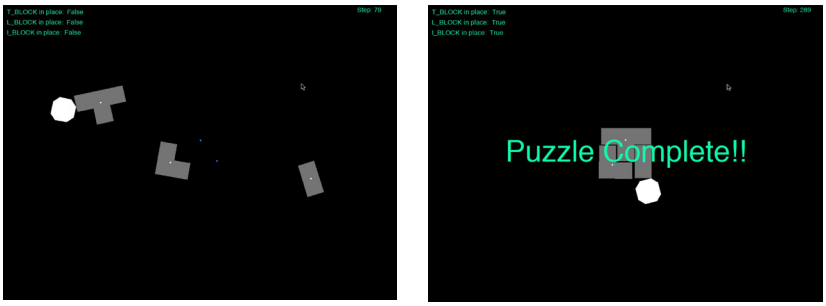


Figure 13.
Manually controlled environment to test the controls and physics implementation.

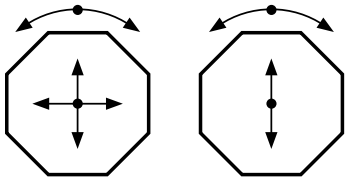


Figure 14.
Diagram showing the degrees of motion of holonomic and non-holonomic control for the agents.

degrees of freedom (DOF) match the degrees of motion (DOM), while non-holonomic control is when the DOF is greater than the DOM (Ben-Ari & Mondada, 2018). A common example of holonomic control is a cart on casters; the cart can freely rotate and move in the x and y-axes. A car, only having two DOM, is an example of non-holonomic control. It can move forward and backward and can turn, but it cannot move sideways. Fig. 14 graphically demonstrates the DOM available to the agent with holonomic control and non-holonomic control.

4.4 OPENAI GYM FRAMEWORK + CHOOSING AN RL ALGORITHM

After testing the manually controlled simulation implemented with PyBox2D and PyGame, I moved on to build a environment to be controlled by a reinforcement learning algorithm. As discussed previously, reinforcement learning is a machine learning approach whereby an agent (or multiple agents) exists and acts in an environment. At each timestep, the agent chooses an action to perform in the environment, which returns a reward and an observation of the current state of the environment (Sutton & Barto, 2017).

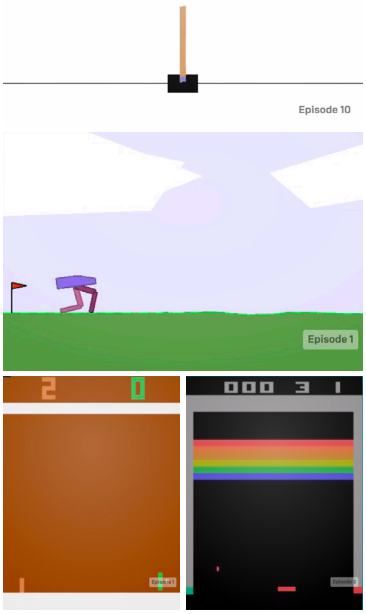


Figure 15.
OpenAI gym environments: (A) Cartpole; (B) 2D Walker; (C) Atari Pong; (D) Atari Breaker

In 2016, OpenAI released OpenAI Gym, "a toolkit for reinforcement learning research that contains a diverse collection of tasks (called environments) with a common interface" (Brockman et al., 2016). The aim was to present a collection of benchmark tasks (Fig. 15) that all researchers had access to and could easily compare results. The common interface that Brockman et al. described is the abstracted structure that a researcher interacts with each environment, irrelevant of the type (classic control, algorithmic, Atari, pybox2D, MuJoCo, etc.).

Each environment class contains the following methods: step, reset, render,

close, and seed. At the end of each episode, the `reset` method resets the environment to its initial state. The step method takes the action as input and returns: the agent's `observation`; the `reward` for the previous action and transition into the current state; a boolean `done` indicating whether the episode is complete; and an `info` dictionary with optional diagnostic information. See Table 2 for sample code that shows the initialization of the environment, Cartpole-v0, and twenty episodes where the agent takes a random action for each of the 100 steps per episode. If `render` is called, then a graphical window will display the agent acting in its environment, otherwise the calculations will run behind the scenes. Rendering is computationally expensive, so it is best to avoid using except with testing or debugging.

Table 2.
Example code from simple cartpole environment. Edited excerpt from OpenAI gym documentation

```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample() # random action
        observation, reward, done, info = env.step(action)
```

To develop my own environment, I used OpenAI's "box2d" environments as a framework for adapting my manually controlled simulation to work within the OpenAI Gym structure. By following their example and maintaining the common interface, I could use existing algorithms and documentation specifically for OpenAI Gym. To assess the success of my environment design, I determined that rather than implement my own version of an RL algorithm, I would use one of OpenAI's baseline implementations. Starting in 2017, OpenAI began open-sourcing baseline implementations of well-known algorithms based on best-practices in order to "make sure that apparent RL advances never are due to comparison with buggy or untuned versions of existing algorithms" (Sidor & Schulman, 2017, p.1). By using one of OpenAI's implementations, I could adapt the code for my own use and ensure that there were no bugs in the unedited implementation. Keeping the algorithm constant, I could evaluate the success of my simulated environment.

Next, I chose an RL algorithm to test my environment. The task at hand helps to determine which algorithm to choose. As discussed in Section 2.3, the observation and action space can either be discrete or continuous, meaning it can either be clearly separate and distinct with a fixed number of observations or actions (e.g. left, right, up, and down) or it can operate within a continuous range (e.g. linear velocity). I chose OpenAI's baseline implementation of Deep Deterministic Policy Gradient (DDPG) with parameter noise (Plappert et al., 2017). This algorithm is ideal for my research because it is can operate over continuous action space, ideal for robotic control, as well as both high and low dimensional continuous observation space.

It was important for the algorithm to accommodate both low and high-dimensional observation space because my end-goal was to transition from low-dimensional observation data, such as location, rotation, and linear velocity, to high-dimensional observation data, such as raw pixels, i.e. images. In using raw pixels as observations, the algorithm takes in direct data on each pixel, typically 3 channels for RGB or 1 channel for grayscale.

4.5 TROUBLESHOOTING

My first implementation (Fig. 16) was composed of a single agent and the same three rectilinear blocks that I designed in Section 4.1 (Fig. 9). The main decisions that I made were about the action space, observation space and reward structure. In this iteration, I decided to discretize the action space into eight actions: North, Northeast, East, Southeast, South, etc.. The observation description contained global location and rotation information about the agent and each block. In addition, I tracked how many blocks were in their final location.

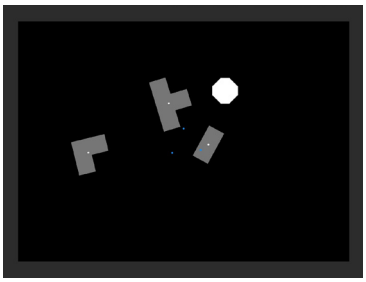


Figure 16.
First iteration of environment: RobotPuzzle-v0. It contains a single agent with three unique blocks that form a square. The agent is tasked with moving the blocks to their specified location, demarcated by small blue dots near the center of the screen.

The reward structure was composed of three main parts: a living penalty, block reward, and puzzle completion reward. Because of the complexity of the problem, I decided to use reward shaping for the living penalty to give the agent hints toward good observation-action pairs. The living penalty, awarded at each timestep, was calculated as the distance between each block and its final location. The further the block was away from its final location, the higher the penalty. This was intended to motivate the agent

to move the blocks closer to their final location quickly.

A reward of +10 was awarded for each block moved into its final position, and -10 was deducted if the block was moved out of place. Once the puzzle was completed, an award of +1000 would be given. Unfortunately, no positive rewards were ever given. Upon training and testing, the agent failed to learn any useful behavior.

In order to troubleshoot the environment, I simplified the simulation to include a single block and no agent. At the beginning of each episode, the block is initialized at a random location and rotation. I changed the action space to be continuous and control the linear velocity (x, y) and angular velocity of the block. I also updated the observation description to contain the block's relative location and rotation information as well as its distance away from its final location.

Lastly, I discretized the reward structure and provided positive reinforcement for moving towards the final goal. I weighed location-based rewards heavier than rotation-based rewards. At each timestep, the block was rewarded +1 for moving towards the goal, -5 for moving away, and -3 for not moving at all. In addition, small rewards of +/- 0.5 was given for rotating towards/away from the final position, with a final reward of +1000 for moving into place. These adjustments proved successful. By simplifying the environment and maintaining a consistent training algorithm, I was able to improve my design and create a trainable environment. Initializing from any random location and rotation, the block learned to move itself to the specified location and rotation within a fine margin of error (Fig. 17).

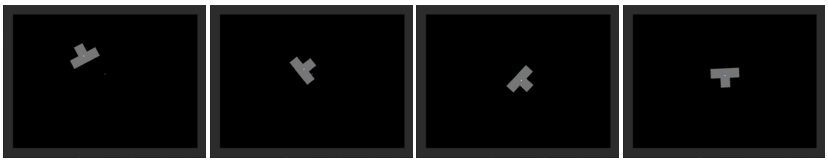


Figure 17.
Simplified environment of block learning to move itself in place.

4.6 CNN VS. LOW-DIM OBSERVATION SPACE

At this point, I decided it was time to see if I could use pixels as input. The most common approach to learning directly from pixels, also called "end-to-end" learning, is to use convolutional neural networks (ConvNets or CNN). CNNs are similar to neural networks, but they assume their

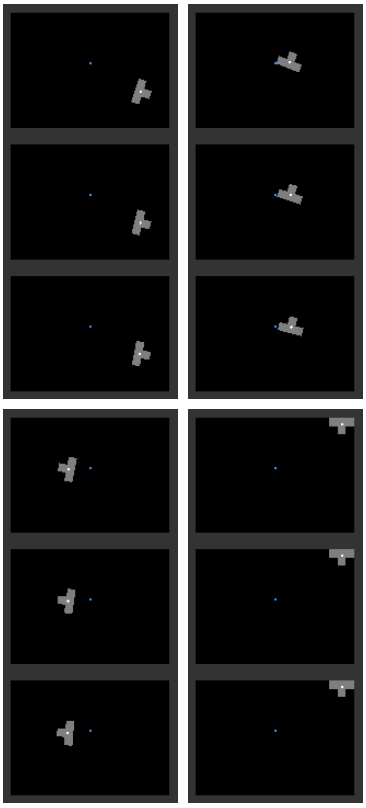


Figure 18.
Observations extracted from the high-dimensional input to the CNN. Each observation contains a sequence of three renderings captured from three different timesteps to capture motion. In this way, the NN can infer motion.

input is an image. They are typically composed of three different types of layers: a convolutional layer, pooling layer and fully-connected layer ***(cite)***. The convolutional layer is the workhorse of the CNN and is essentially responsible for learning a set of filters typically used to identify relevant features for the particular application. The number of filters matches the depth of the convolutional layer.[13] Using OpenAI's CNN implementation drawn from the Deep Q-Network (DQN), I added three convolutional layers to the beginning of both the actor and critic networks of the DDPG baseline.

In addition to adding these convolutional layers, I took a number of important steps to improve the network as outlined in Lillicrap et al. (2015). First, to minimize the number of parameters, I scaled down rendered image to a manageable size for the network, 160x120x3 (width x height x RGB depth),[14] and converted from an 8-bit RGB (i.e. an integer between 0 and 255) into a floating point number between 0 and 1, inclusive. For the network to understand motion, I evaluated two approaches: taking the difference between two frames (Karpathy, 2016) or feeding the network a series of images (Lillicrap et al., 2015; Mnih et al., 2013; Plappert et al., 2017). I chose to send the network a series of images, or observations, as this was the approach documented by Lillicrap et al. (2015) and Mnih et al. (2013). Fig. 18 shows four such observations sent to the network. Lastly, I employed a frame-skipping technique (Lillicrap et al., 2015; Mnih et al., 2013), where the agent only sees an observation and chooses an action every n^{th} timestep, and the action is repeated for the intermediary steps. This minimizes computation and essentially allows the agent to act n times more in the same amount of time.

After setting everything up and running some tests, I set the network to train for a few days. The network was computing between 0.5 and 1.0 steps a second, compared to approximately 160 steps a second with low dimensional data. This occurred for two reasons: (1) rendering the environment rather than calculating everything behind the scenes is much slower; (2) CNNs are computationally expensive and can significantly increase training time. In the end, I decided to stick with low-dimensional observations because of the speed of prototyping and training. Learning directly from raw sensory data is a well-established research challenge, especially in robotic

[13]
For more information about CNN or NN, refer to Section 2.3 or to the learning resources in the appendix.

[14]
If you are using a computer with a high-resolution screen or retina screen (mac), you may input 160x120 as your screen size, but receive a vector twice the size. In that case, you can either downsample the image or input the dimensions as half the size that you want.

control, and while it is a desirable facet of research, it is not necessary for the scope of this thesis.

4.7 IMPROVING THE ENVIRONMENT WITH DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

After deciding to focus on low-dimensional observations, I updated my environment to include a single agent with one block. Now, the agent took the actions in the environment, rather than the block. The agent's action space was continuous and controlled the linear velocity (x, y) and angular velocity of the block. This gave the agent holonomic control, where the degrees of freedom matched the degrees of motion. The majority of the following tests involve adjusting the observation description and reward function to incentivize the agent to learn an optimal policy. It is always import to test this because the agent will act to maximize rewards, which may or may not be what the designer intended.

Moving from images to low-dimensional observational data, I needed to capture information that would have been evident from images, such as the block shape. At each timestep, the observation included the absolute location and rotation of the agent, the agent's relative location to the block, the block's relative location and rotation to the final location, and a boolean that represented the agent's contact with the block. To represent the shape of the block, I included the absolute location of each boundary vertex.

I discretized the reward function by assigning a fixed reward/penalty for the block moving towards its final position, away from it or not at all. I did the same for the block rotational position and the agent's position relative to the block. In addition, I gave a small reward for contact with the block and a large penalty when the agent was in contact with the wall.

After training, the agent learned to interact with the block but was not able to move it towards the goal (Fig. 19). Because the agent started at the center of the environment everytime, it had trouble learning to first move to the other side of the block. Instead, it typically pushed the block to the wall, then would push against the block while rotating in an attempt to get behind it. Overall, it appeared to be a challenge for the single agent to have control manipulating the block.

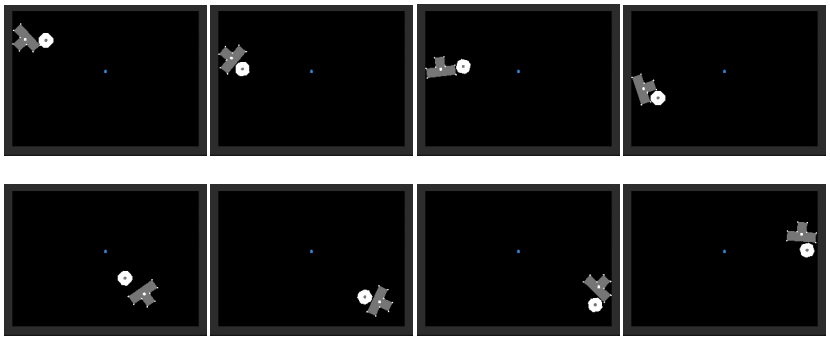


Figure 19.
Top and bottom rows are two separate episodes of the agent manipulating the block. As mentioned, it pushes the block to the wall then attempts to move it away by turning its chassis.

IMPROVING THE REWARD FUNCTION

At this point, I made important changes to the reward function to improve the behavior. First, I changed the living penalty to the sum of the distance between the agent and the block, `agent_dist`, and the distance between the block and its goal position, `block_dist` (Fig. 20). After testing this reward function without any additional weights, I observed that the agent did not learn any ideal behavior. I then added weights to the each distance in order to give priority to the distance between the block and its goal, `block_dist`, over the distance from the agent to the block, `agent_dist`. I thought that because this distance was more important that I needed to give it a higher weight. Unfortunately, the agent did not learn any useful behavior, so I switched the weights giving a higher weight to the distance between the agent and the block, `agent_dist`. This change resulted in the agent successfully interacting with the block. Upon reflection, when dealing with a multi-step task, it makes sense to weigh the earlier step heavier than the following steps as the agent needs to complete it first.

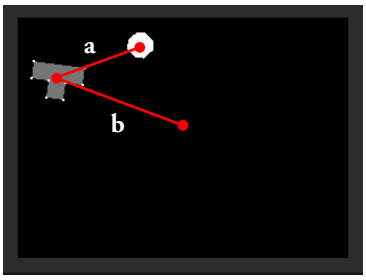


Figure 20.
Diagram illustrating the two distances used for reward shaping: (a) agent_dist; (b) block_dist

In addition, I also added a reward based on change in distance for both `agent_dist` (`agent_delta`) and `block_dist` (`block_delta`). This reward is negative when moving away from its goal and positive for moving towards it. I added weights to these as well, but this time gave a higher weight to the `block_delta` because overall these rewards are smaller than the distance based rewards. This addition seemed to improve the behavior.

ADDING A SOFT CONTACT

Another challenge in this task is that the agent may not know which direction to move. When the observation space is too large and the agent is constantly initializing in random locations, it may never reach its gaol

state to receive the final reward. One way to help is to implement a soft contact between the agent and the block. Soft contact is when the agent constantly applies a small force on the block, even when it is far away, giving the agent a sense of which direction to move. I manually implemented soft contact as a linear force based on the distance from each agent to the block. I used an exponential function, `force = base**(-agent_dist)`, for a quick fall-off, and experimented with the base until it seemed to be appropriate.

TESTING A NEW ALGORITHM

To see if another algorithm may work better, I tested Proximal Policy Optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017). The experiments began with promising results around 1M timesteps, then after training for an additional 5 million timesteps, the magnitude of agent's actions would decrease to be so small that it would barely move at all. By recording and plotting the variance of the actions over each episode, I observed that the variance drastically decreased over time. The training time was much faster than DDPG, so it could be used to test out different parameters in the reward function, but not for any extensive training.

Concurrently, I decided to see if the agent could learn a much simpler toy task. For each episode, I placed the agent in a fixed location with the block in between the agent and the goal. After approximately 1 M timesteps, it was easily able to learn an optimal policy to move the block to the goal. This was simple to learn because as soon as it learned that it needed to move up a specific way to push the block into place, it could easily repeat it. It was consistently starting in the same position, so there was less variability in the observation. Unfortunately, this toy task is too simple and produced a policy that could not generalize to any other initialization location.

MULTIPLE AGENTS + SIMPLIFYING THE TASK

Given the scale of the environment and the fact that a single agent seemed to be struggling to manipulate the block alone, I decided to add another agent to the environment and experiment with their interaction. It was an ideal time to see what collaborative behavior would emerge between the two agents. In this scenario, even though I included multiple agents, they were both centrally controlled by the same learning algorithm. There

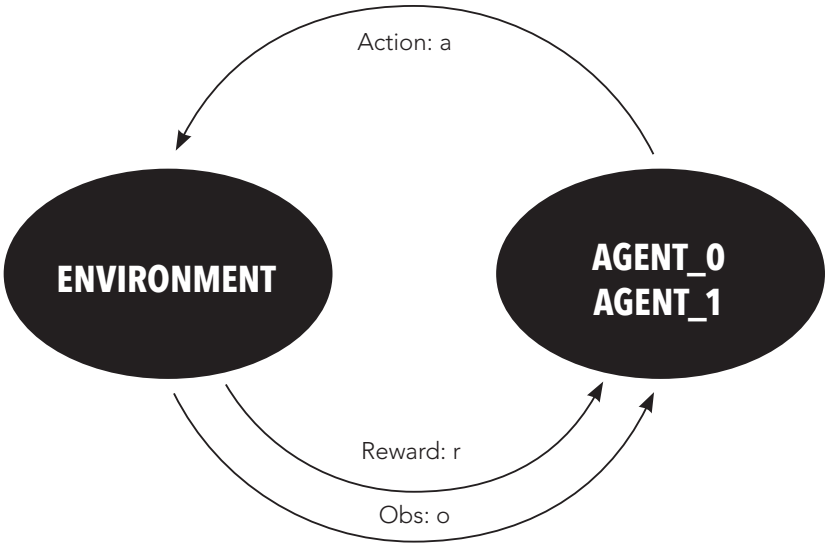


Figure 21.
Reinforcement learning diagram showing how the centrally controlled agents receive the same observation and reward.

was a single observation and a single reward function and the action space expanded to include controls for both agents (Fig. 21).

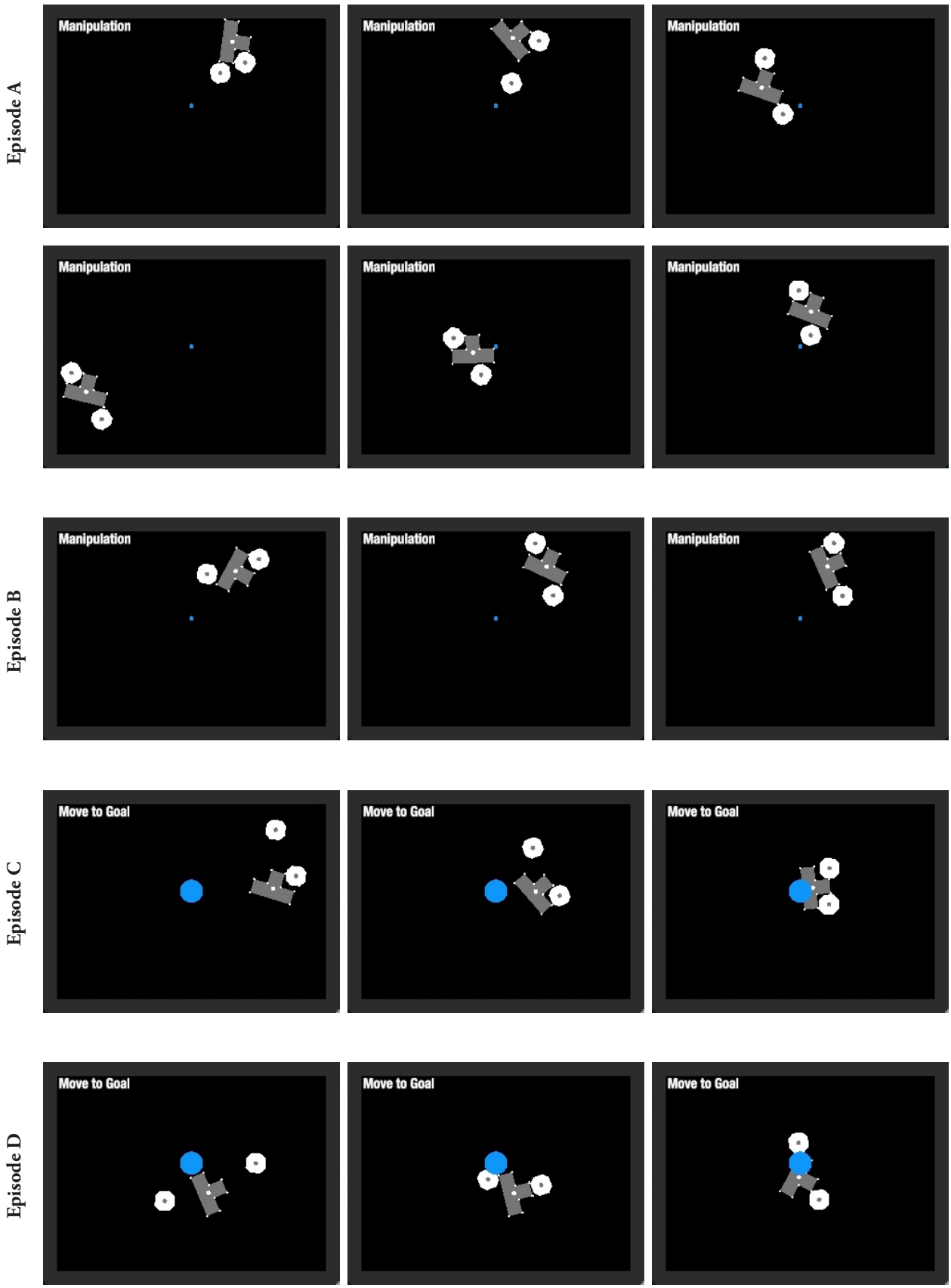
The observation included information about each agent and the goal block. Each agent's global location and rotation, relative location to the goal block, linear and angular velocity, and contact with the block were all documented. In addition, the observation included the goal block's relative location and rotation to the goal location and the global location of the block's vertices.

After seeing the agents develop some collaborative behavior for manipulating the block, but fail to successfully move it to the goal location, I determined that the goal was too specific for the agents to be successful early on. I decided to make the problem easier by making the margin of error larger and removing rotation from the requirement to *be in place*. After implementing these changes, the agents began successfully moving the blocks in place (Fig. 22). Some specific behaviors that emerged were turning the block together, taking turns to push the block, and guiding from different angles.

Figure 22.
(Opposite) Series of screenshots from testing episodes show collaborative behaviors agents learned for moving the block, such as rotating, pinching, and guiding.

PREPARING SIMULATION TO MATCH REAL WORLD

In my near-term next steps, I aim to transfer this learning from simulation to physical robots. While there are many challenges in transferring learning from simulation to real-world, there are some steps I took to prepare the simulation for transfer. First, it was important to make sure that the


Episode A
Manipulation
Manipulation
Manipulation
Manipulation
Manipulation
Manipulation
Episode B
Manipulation
Manipulation
Manipulation
Episode C
Move to Goal
Move to Goal
Move to Goal
Episode D
Move to Goal
Move to Goal
Move to Goal

simulated screen would match the overhead camera. The proportions needed to match, as well as the Cartesian coordinate system and dimensions. I normalized all of the distances so that the width ranged from [0, 1] and the height was scaled respectively. I maintained soft forces for initial learning and decreased the strength of the force over time.

I simplified the environment so that the agents always initialized in the left third, the block in the middle third and the goal in the right third. (Fig. 23) I also made the margin of error very large at the beginning of each training session and decrease in size over time. This allowed for easy wins towards the beginning, with the task becoming more difficult as the training progressed. I removed the walls and assigned a large penalty if the agent moves out of bounds and a smaller one if the agent pushes the block out of bounds. Both of these penalties decay over time so that if they agent moved out of bounds in the last few timesteps it received a smaller penalty than if it did in the first few timesteps. The rewards for each episode's terminal state typically ranged from lowest score to highest in this order: agent moved out of bounds, agent stayed in bounds (often going in circles), agent pushed block out of bounds, and agent pushed block to goal.

The reward was based on four changes in the state: distance of agent to block, `agent_dist`; change of distance at each timestep between agent and block, `agent_delta`; distance of block to goal, `block_dist`; delta distance between block and goal, `block_delta`. I ran a series of tests where I changed the weight for each variable of the reward function, i.e. [25, 50, 100] for delta distances and [0.1, 0.25, 0.5, 1.0] for distances. I initially tried to

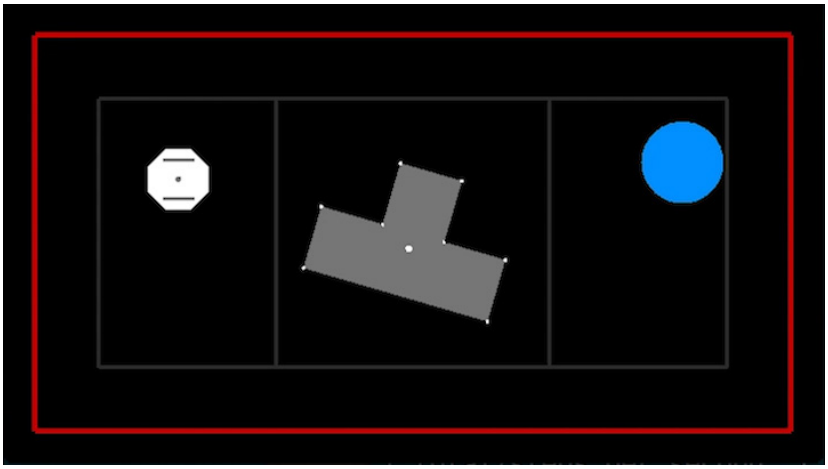


Figure 23.
In oder to better match the real world, the updated environment does not have boundary walls and the agent has non-holonomic control, similar to the way the physical robots move.

evaluate each test based on the reward or number of timesteps, but the plot of the reward history fluctuated significantly and purely increasing the timesteps may mean that the agent is learning to go around in circles without producing any meaningful behavior. Instead, I chose which parameters to test further by running each model without any parameter noise for 5 epochs and evaluating the behavior visually. Concluding those tests, the best parameters seemed to be 25 (`agent_delta`), 0.25 (`agent_dist`), 100 (`block_delta`), 0.0 (`block_dist`).

To improve performance further, I will need to tune the hyper-parameters of the network, specifically the learning rate of the actor and critic networks and the parameter noise. Using a randomized approach for choosing hyper-parameters (Bergstra & Bengio, 2012), I will take the top two parameter settings for the reward functions and test the each environment ten times with randomly initialized hyper-parameters.

SEEING AS THE AGENT SEES

To understand what the agent "sees," I experimented with some visualization tweaks to the renderer (Fig. 24). The agent receives its observation as an array of numbers. There are no labels or background information for the agent to understand what the numbers mean. It learns purely through interacting with the environment and receiving rewards based on its actions. To give a sense of what the agent "sees," I removed extraneous information such as boundaries and filled shapes and included only points and relational information. In this example, observational data includes the global location and rotation of each agent and the block's vertices. It also includes the relative location between each agent and the block and between the block and the goal–represented by the dashed line. The large circle around the goal represents the margin of error allowed for completing the puzzle. This episode is early in training, so the margin of error is significantly large.

4.8 MOVING TO A DECENTRALIZED APPROACH: MADDPG

Because my main goal is collaborative behavior learned through interaction between agents, I believed that using a centrally controlled learning algorithm was not as true to the theory as necessary. As a result, I decided to test OpenAI's multi-agent deep deterministic policy gradient

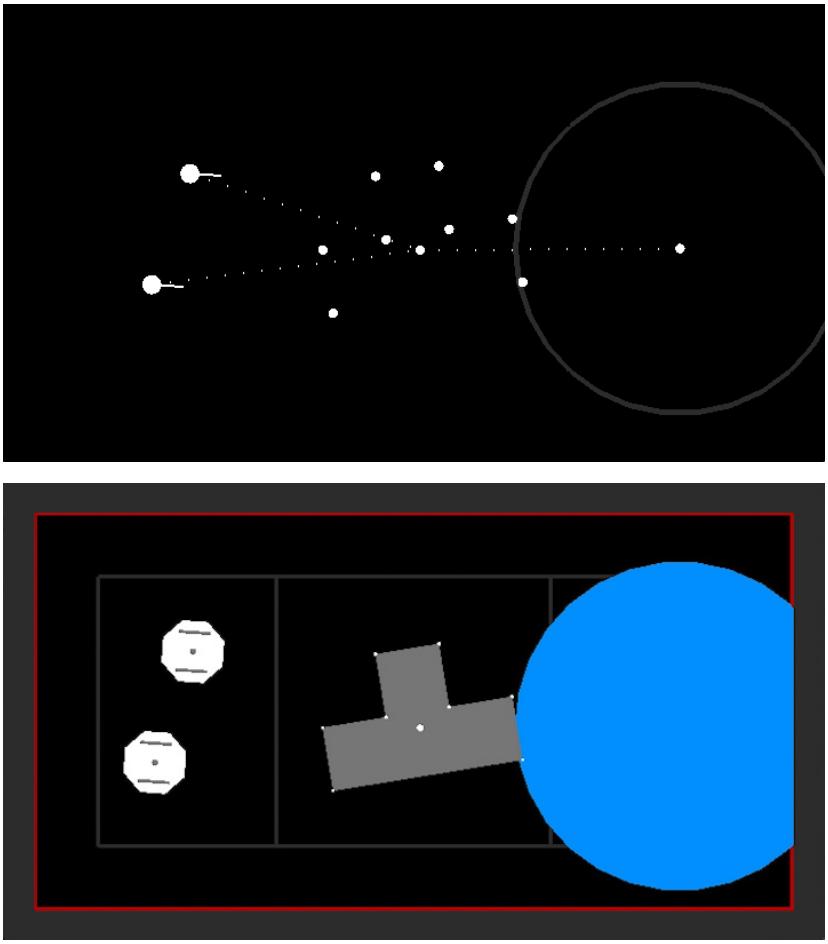


Figure 24.
The two images are snapshots from the same simulation with different rendering techniques. The top more similarly represents what the agent might see, while the bottom is a full representation of the objects in the environment that people can easily perceive.

(MADDPG) implementation with separately trained agents. In order to use this implementation, I needed to change my environment design to consist of separately controlled agents with their own unique observations (Fig. 25).

As described in Section 2.3, OpenAI developed eight multi-agent particle environment with a variety of competitive and cooperative tasks to test MADDPG. These particle environments were simple enough that the researchers implemented them with their own physics calculations. I utilized their framework to program a new environment using pyBox2D for a physics engine, contributing a new environment framework to the multi-agent gym environments. As in the previous DDPG controlled multi-agent environment, the agent's controls are non-holonomic, meaning it cannot move sideways. The actions adjust the linear and angular forces, allowing it to drive around similar to the physical robots. In addition, I added the boundary walls back to this simulation to avoid the agents moving out of

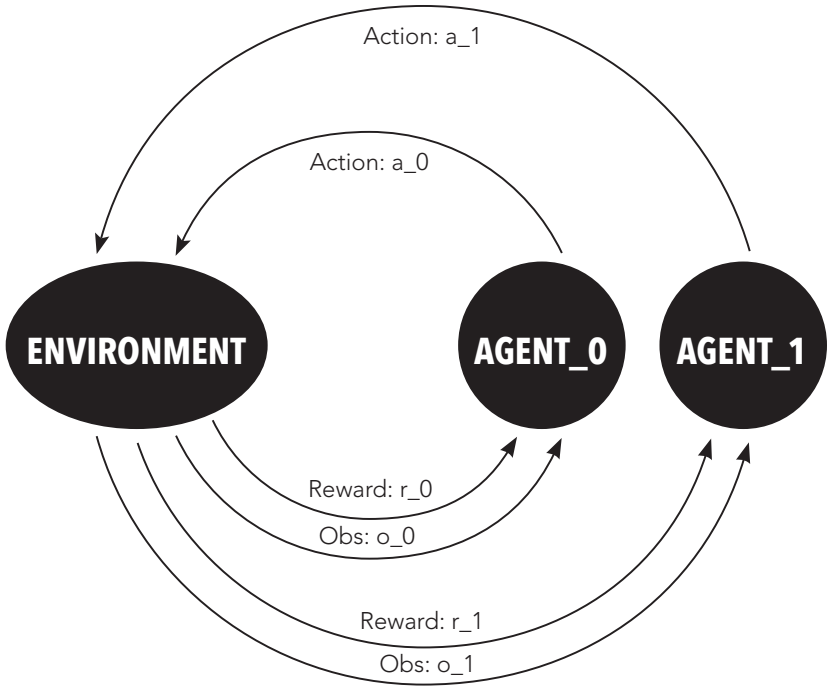


Figure 25.
Reinforcement learning diagram showing how in a decentralized multi-agent environment, agents receive the their own observation and reward and take their own actions.

the frame and ending the episode early. Lastly, I made the margin of error larger, so that it was easier for the agents to move the block into place.

I used a continuous action space between [-1, +1] to control the linear and angular forces controlling the robot. This was multiplied by a maximum force of 0.25. After training for 1M, the agent did not seem to learn anything. I hypothesized that it was because the agent was not exploring far enough to be able to find any success to guide it. On my next test, I increased the maximum force to 1.0. After training approximately 1M timesteps, the agents did learn to move the blocks in place, but the MADDPG implementation allowed the agents to choose actions between [-120, +120].

Upon closer evaluation of the code, I observed that the actions were selected from a Gaussian distribution. This means that while larger numbers are probabilistically unlikely, it is possible for them to be selected, with increasing likelihood especially if those actions repeatedly result in high rewards. I wanted to keep the actions restricted to be between [-1, +1], so I clipped the selected actions to be between [-1, +1]. This eventually resulted in returning "not a number" or NaN for the state, reward, and action. Further work is needed to develop a successful and controlled MADDPG implementation.

4.9 HOLONOMIC VS. NON-HOLONOMIC CONTROL

After running numerous tests in environments with holonomic control (using DDPG) and non-holonomic control (using DDPG and MADDPG), I determined that the control can have drastic influence on the success of the agent in an environment. The agents with holonomic control developed more complex collaborative behaviors than agents with non-holonomic control (Fig. 26). When the agents had non-holonomic control, i.e. moved similar to a car, they needed to initialize in the simplest setup in order to reliably move the block successfully. This simple initialization mandated that the agents always appeared on the left third of the screen facing right, the block appeared in the middle third, and the goal on the right. If the agents initialized facing a random direction or in a random location, then they would not be able to find a solution.

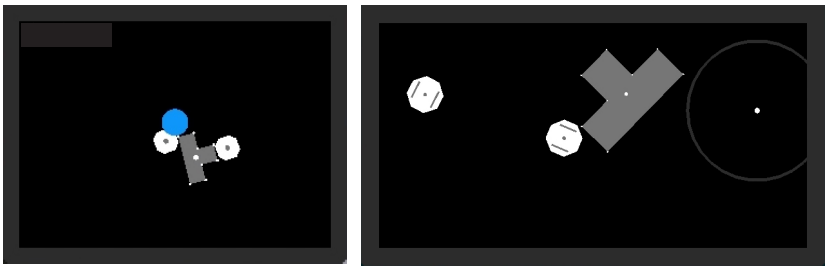


Figure 26.
(Left) Holonomic controlled environment. (Right) Non-holonomic controlled environment.

This appeared to be because the agents needed to be facing the correct direction before they could successfully move the block. The non-holonomic control added significant complexity to the problem. In contrast, with holonomic control, it was easier for the agents to explore the state space because they could freely move in any direction. After making this discovery, I started experimenting more with holonomic control in environments with more agents.

An important component of collaboration is that by working together the agents can achieve more than one could alone. To test this in a more extreme environment, I doubled the size of the block and increased the density of the block so that a single agent could barely move the block. Only by working together could the agents move the block to the goal within the maximum timesteps. I tested this new block with ten agents and then five.

In the environment with ten agents, there were two primary results. The

first, which is to be expected with so many agents, is that the agents got in each other's way and pushed the block to the wall (Fig. 27, left). With all of the agents starting in different directions, they each arrive at different sides of the block and start working against each other. This is what often happens when collaboration goes wrong–too many people are trying to take the lead and, as a result, things often get worse.

The second more surprising result was that the agents started acting like gears and took a mechanical approach to moving the blocks (Fig. 27, right). Because the block was so heavy, the agents could move the block faster by stacking themselves together and rotating like gears. This worked well when the block was near the wall, but was more difficult when it moved further away.

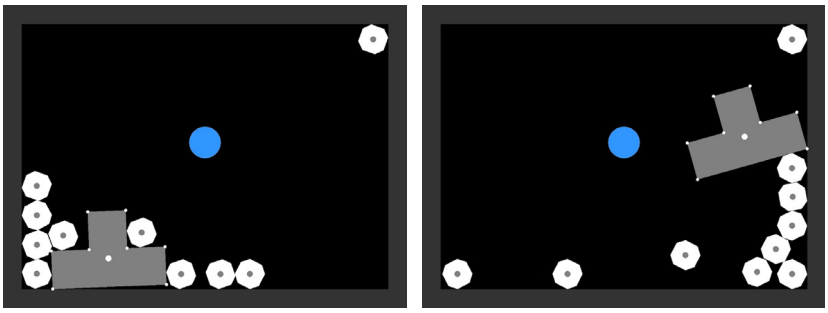


Figure 27.
(Left) When collaboration is too much, all of the agents get in the way of each other. (Right) When collaboration turns mechanical, the agents acts as gears to mechanically move the block.

In the final simulation, I tested five agents with the same large block. This number appeared to be just right where some agents collaborated to move the block, while others went off into corners. Overall, they did not get in each other's way as much as ten agents did. In Fig. 28, the agents use the mechanical gear strategy to move the block away from the wall, then they continue to push on different parts of the block before moving into a linear formation for the final push.

4.10 FRAMEWORK FOR SIMULATION-TO-REAL-WORLD TRANSFER

The last phase is to transfer the policy from simulation to real world. As described in Section 2.3, learning transfer is a non-trivial task where many issues can arise. In this section, I lay out a framework for learning transfer with many of the components implemented separately, but not yet combined into a complete system.

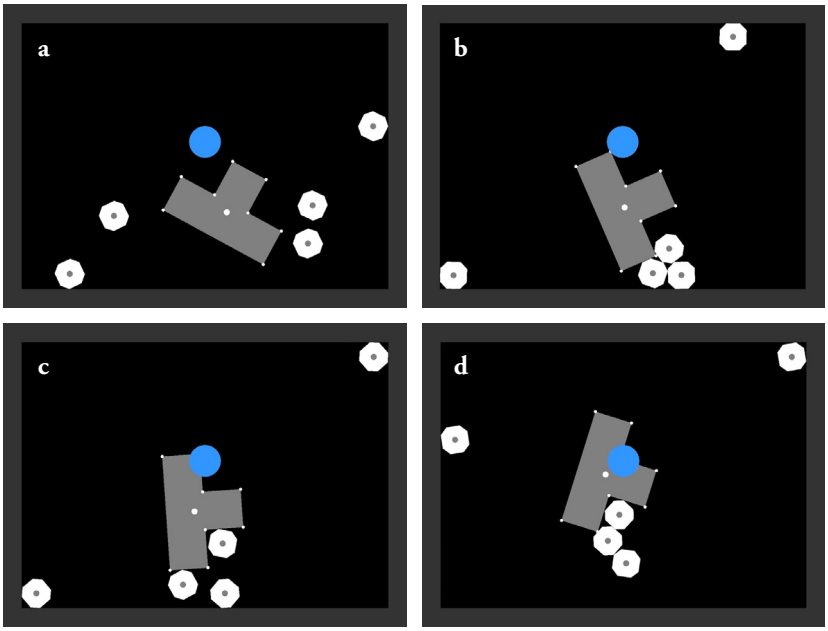


Figure 28.
An environment including five agents and one heavy block. Three of the agents collaborate to move the block to the goal.

The most important step is to match the observation description from the simulation to the real-world. As discussed in Section 4.7, I took initial steps to prepare the simulation to match the physical world. To calculate the relative position of elements in the observation description, I needed to be able to extract the global location and rotation information of each agent and the block as well as the block's vertex locations. To gather this data, I installed an overhead camera and used OpenCV, a computer vision package, to identify various objects.

For information that is more difficult to gather accurately from an overhead camera, such as linear velocity, angular velocity, and contact, I can first test to see if they are absolutely necessary for the algorithm to find an optimal policy. If so, then I can slowly add noise to those data points as the algorithm learns. This allows the algorithm to use the information while initially learning but progressively rely less and less on that data. Another way to account for contact would be to install ultrasonic sensors on each side of the robot. When the sensor measures an object under a minimum distance, it can record that it is in contact with that object.

Using color recognition and contour detection, I extracted specific shapes to identify the block and robots' information. First, I marked the blocks with different colored triangles. The system can identify each block based on the color of their triangle, as well as their location and rotation based

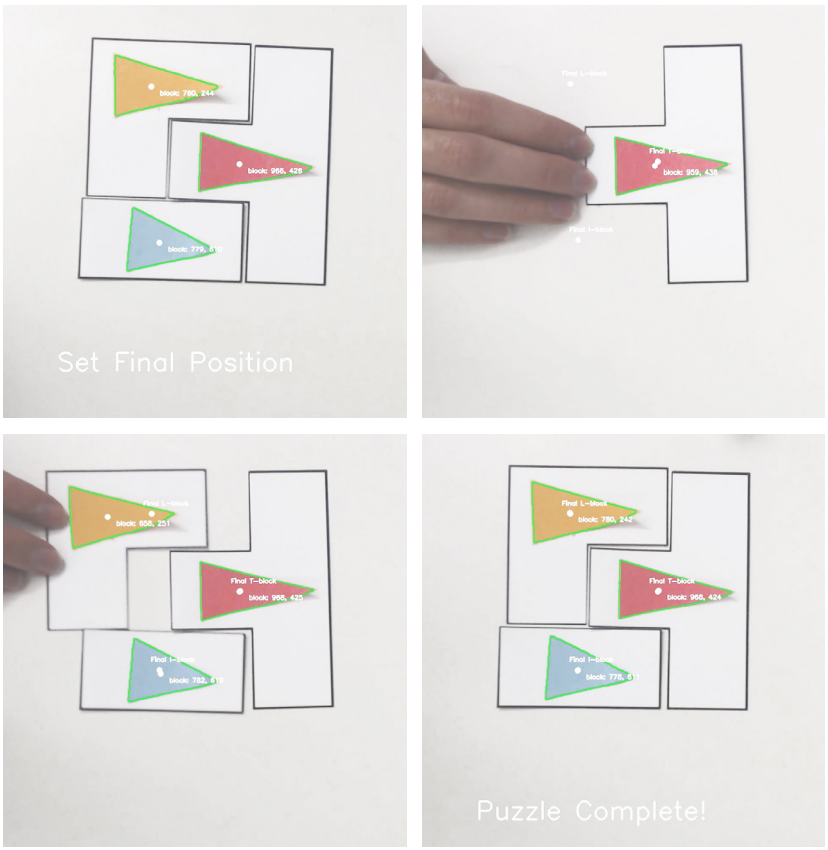


Figure 29.
The vision system can identify each block's location and orientation using the triangle. The user sets the blocks' final positions and the program can identify when the pieces are in place.

on the orientation of the isosceles triangle (Fig. 29). I developed a program that would allow users to set the final position simply by moving blocks into place and pressing a key. The program would then check that each block was in place before declaring the puzzle complete.

Color recognition and contour detection work best when there are clear boundaries between shapes and colors. To create a plain background, I used thick black paper as a non-stick and non-reflective surface for the work area. I painted the blocks white so they would stand out against the black surface. The new robots, Builder Robot 2.0, are made out of black acrylic and sanded down to create a matte to avoid hot spots on reflective surfaces.

The components of Builder Robot 2.0 (Fig. 30) are the same as version 1.0, but the chassis and control system have been updated. The chassis now has an enclosed area for the batteries, Raspberry Pi, and motor hat to sit where the wires are contained. As I am using an overhead camera to identify the robotic agents, it's important to hide any extraneous objects that may be visually distracting the recognition system. Each robot has its own colored

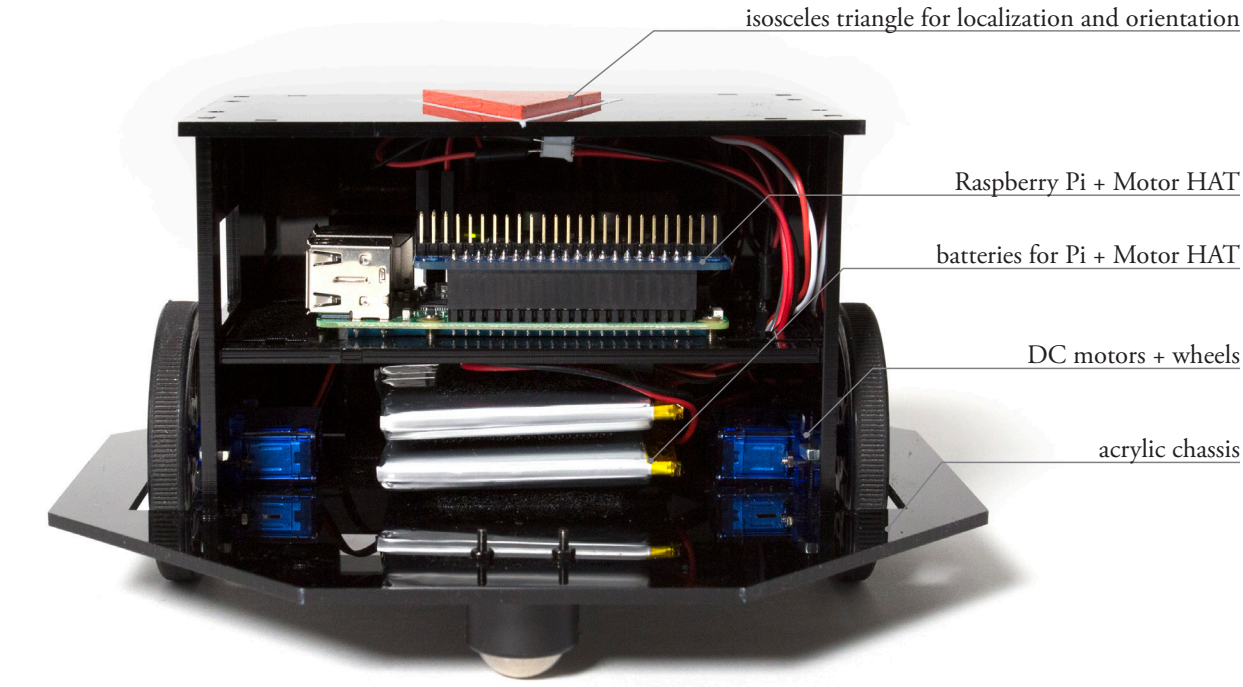


Figure 30.
Robot Builder 2.0 is built with the same components as Robot Builder 1.0, but it has an updated chassis for being identified with computer vision.

marker, an isosceles triangle pointing forward with its centroid matching the robot's centroid. This allows for easy tracking of the robot's location and rotation. As for the control of the robot, the DC motors do not have consistent power from zero to full speed. At low speeds, the motors do not turn at all. To fix this, I tested the motors to determine at what speed they started moving. I remapped all speeds to be between the new minimum and full speed, rather than zero.

To track the block, I initially placed another color-coded isosceles triangle at the center of the block, matching the centroid position of the triangle to that of the block while pointing up. As with the robots, this tracked the position of the centroid and rotation of the robot, but it did not account for the block's vertices. Instead, I used a similar approach to identify the boundary of the block and extract its main vertices. As the block rotated, the order of the vertices kept shifting. It is mandatory to maintain the same order throughout all testing. To keep track of the vertices, I rotated to isosceles triangle to point to the starting vertex and sorted the vertices based on that point (Fig. 31).

At this point, I had all of the primary components of the observation description. Next, I tested communication between the computer and

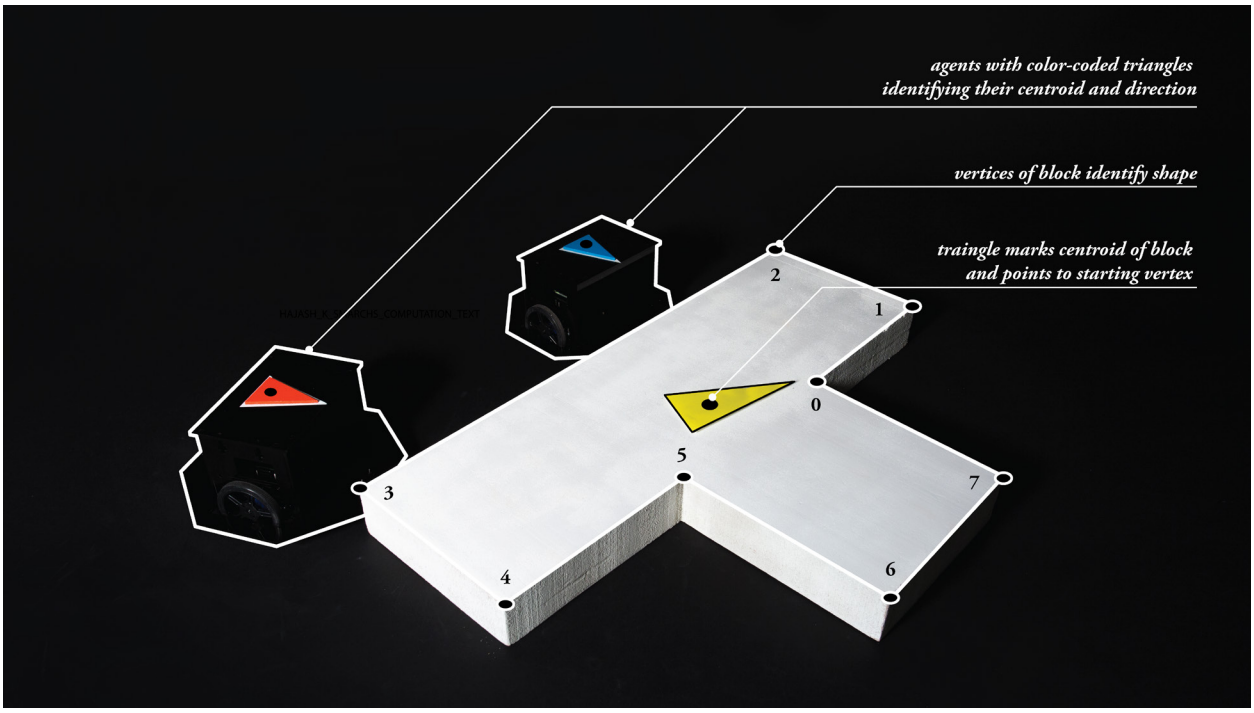


Figure 31.
Diagram of the agents pushing the block, demonstrating the various components of the vision system

the robots using the library, PyOSC. I integrated the program that I had developed for identifying the block and robotic agents with the communication to send instructions to the robots while tracking their location. I began testing the controls of the robot in the real-world and in simulation to evaluate how to match the two environments. The manually-controlled integrated system (Fig. 32) shows the agent identified on the left, the block with all of its points numbered and centroid calculated in the middle, and the final goal position on the right.

The next step is to perform more rigorous system identification to match the simulation to real world as much as possible. Some issues include imprecise tracking of points, inconsistent control of the DC motors, and inconsistent friction that prevents the robot from moving the block in certain positions. To address the point tracking, I will add a small amount of noise to the simulation to better match the constant fluctuation of the points. Rather than use DC motors, I will look into using stepper motors which are much more consistent in speed and control. This may better translate from simulation to real world. Lastly, to deal with inconsistent friction, I will implement a small randomized force that pushes against the agent in simulation.

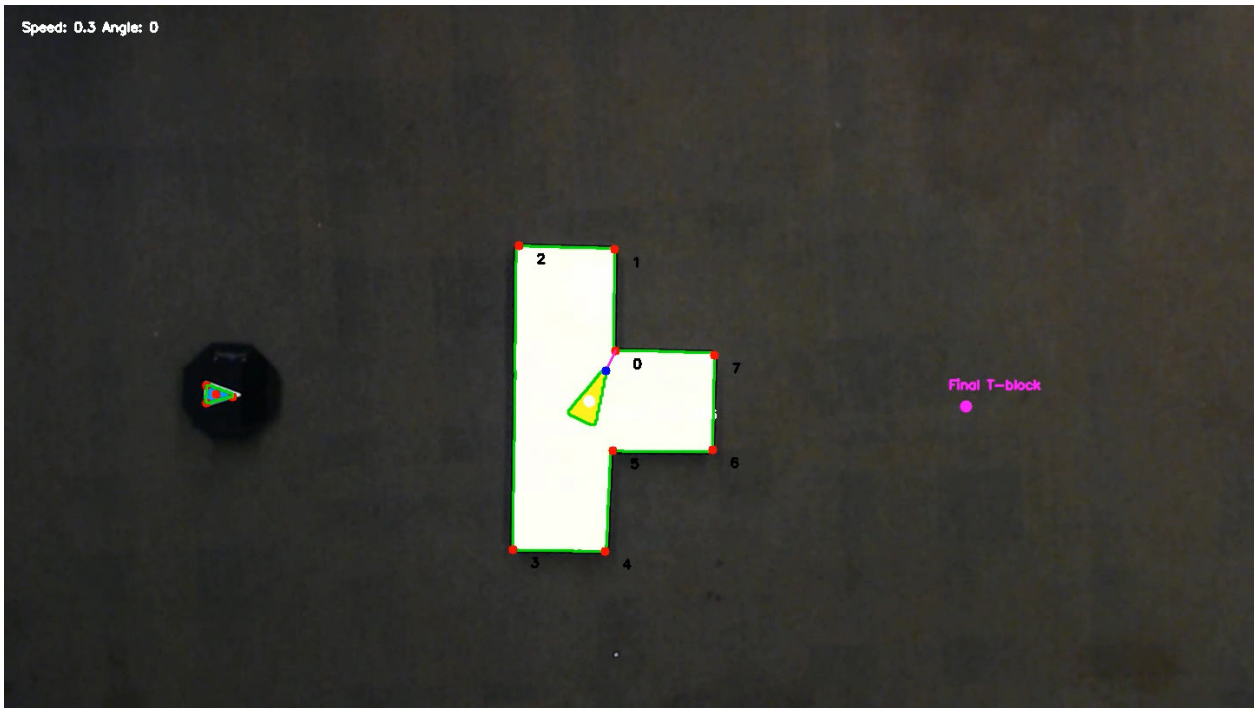


Figure 32.
Computer vision overlay on view from overhead camera

4.11 ANALYSIS + LIMITATIONS

In this section, I use the three phases of robotic collaboration–preconditions, process, and outcomes–as a framework to evaluate this research. As I outlined in Chapter 3, the preconditions include the environmental factors that facilitate collaboration, the process includes the activities that take place during collaboration, and the outcomes are the results and methods for which to evaluate the success of collaboration.

In this research, the ***preconditions*** are the settings of the environment, abilities of the agents, and the shared or individual goals. More specifically, the preconditions include: number of agents; the agent's control system; the agent's strength and observations; the agent's ability to manipulate blocks; the size, density and shape of the blocks; the number of blocks; reward function (shared or individual); environmental settings (friction, damping, etc.); and initialization settings.

I developed nine different environments, or sets of preconditions (Fig. 33), ranging from a single block as the agent to ten agents manipulating the block. Within the environmental conditions, I primarily tested the representation of the observation data, the agent's control system (holonomic or non-holonomic), the initialization settings and the number of agents.

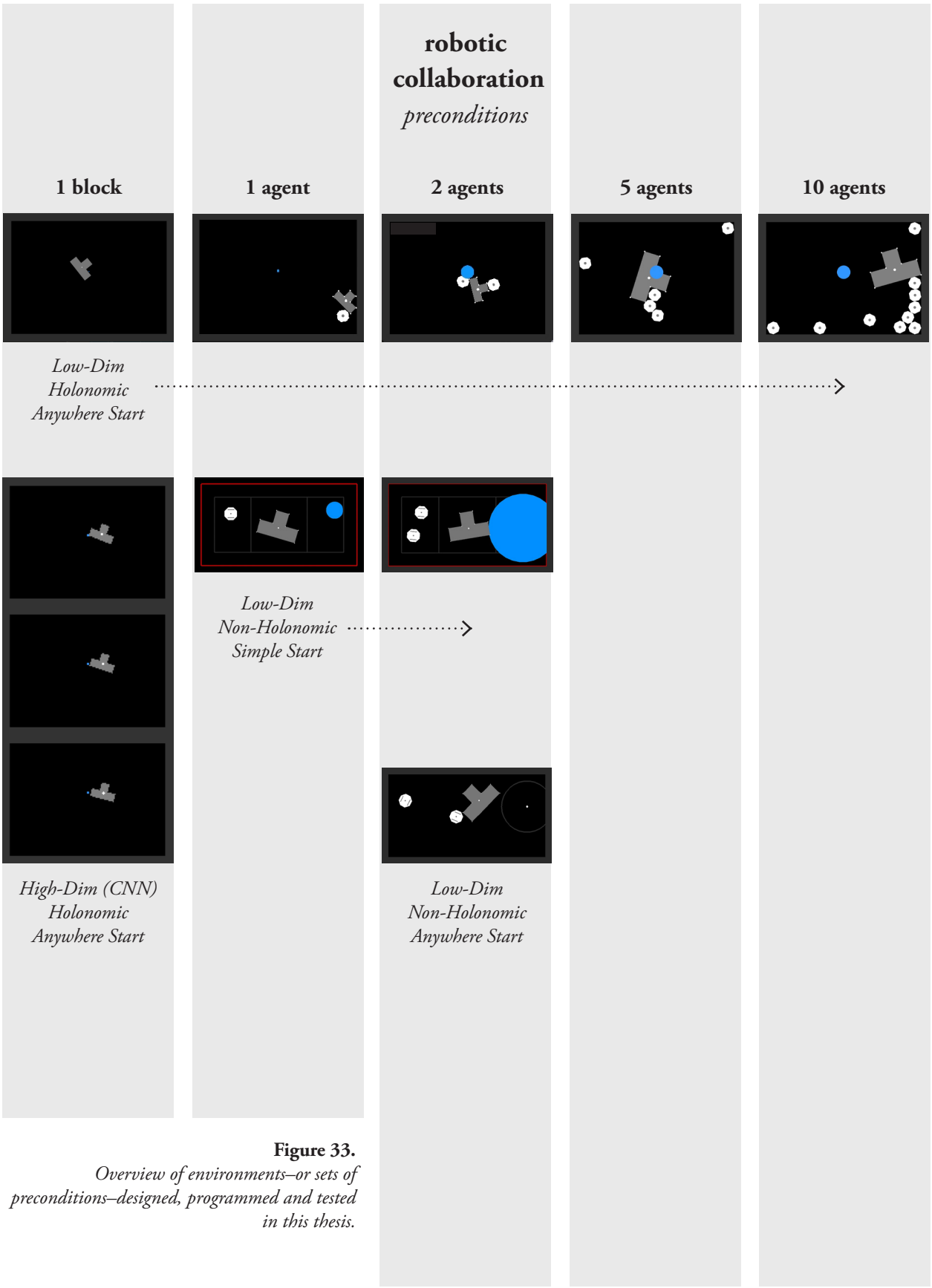


Figure 33.
Overview of environments–or sets of preconditions–designed, programmed and tested in this thesis.

After developing the first simplified environment, I had to decide between low-dimensional observational data (location, rotation, velocity) and high-dimensional data (raw-pixels). Due to the extensive training time for CNN and lack of computing power, I decided to move forward with low-dimensional observational data. While low-dimensional data is acceptable, it is not ideal when working with robots in the physical world when data tends to be noisy and imprecise. With more time and capabilities, I would push for using raw-data as input.

As discussed in Section 4.9, the agent's control system dramatically affects the agent's ability to learn complex collaborative behaviors. If the agent is restricted from fully exploring the space by non-holonomic control, the algorithm may never converge to find an optimal policy. Whereas with holonomic control, the agent is easily able to explore and is much more likely to find a successful path to the goal state through random actions.

In addition, the initialization setting was significantly restricted if the agents had non-holonomic control. This restriction is a important limitation on the current system. In future work, I would switch back to holonomic control and build a new robot with omni-wheels to match the simulated control system.

The most important part of collaboration is the ***process*** through which collaboration occurs. More specifically, the process includes the emergent behaviors that the agents develop through interacting with each other and their environment. In my research, I evaluated success through identifying and analyzing the emergent behaviors . These behaviors were more important than the specific outcomes.

The most interesting collaborative behaviors emerged in the holonomic–based environments. In the non-holonomic and simplified environments, the agents typically learned to move towards the block and push next to one another. If the agents did not land on the correct side of the block or missed the goal, there was typically no recovery.

In contrast, in holonomic multi-agent environments, the agents developed methods for collaboratively rotating the block together and pinching it between them while moving. Because the block was not too heavy, one

agent could take the lead pushing the block towards the goal, while the other agent learned to aid this agent by guiding the block from a different direction. In the environments with a much larger and heavier block, agents learned how to take advantage of their octagonal form and the static walls. By placing themselves next to the walls and with others, the agents learned to act as gears, mechanically pushing the block when their own strength was not enough. Each of these behaviors demonstrate a new way to think about collaborative behavior for moving blocks.

The final ***outcomes*** include whether the agents achieved their goal or not. Through this lens, the only way a collaboration could be successful is for the agents to move the block into place. Unsuccessful collaborations would include: the agents pushing the block around, but not to the specified goal; the agents moving in circles; or the agents going out of bounds or running into the walls. Another way to evaluate the outcomes in a less physical and spatial way is to analyze the total reward through training. If the agents learn to maximize the reward through training, especially if this is correlated to successfully manipulating the block, then this would be deemed successful.

Through this research, I realized how difficult this task was, even after I simplified it to a single block. It is difficult for a number of reasons which I briefly stated at the beginning of this chapter. First, it is a multi-agent environment which means that each agent is both observing the environment–either from its own perspective (MADDPG) or from an overall perspective (DDPG)–and changing the environment through its own actions. This makes it increasingly difficult for the agents to know which actions were important in a successful episode and which were not.

Second, the task requires the agents to manipulate an object. Many reinforcement learning tasks involve the agent learning how to exist in an environment without manipulating anything. The tasks could include walking, landing, flying, driving, or maintaining some level of equilibrium. Rather than only accounting for itself, the agent must interact with another object within the environment, adding more complexity to the task.

Third, the task of randomly initialized agents moving a block in place is a

multi-step task. First the agent has to find the block, then push the block towards the goal. It not only has to learn what to do, but what to do in a specific order. This adds to the issue of credit assignment, where the agent does not know which actions resulted in a win or a loss.

Lastly, this task is difficult because the agent exists in continuous state space. This means that the agent will never be in the exact same state again. Therefore, it is difficult for the agent to look back on its past experience and predict what action to take in an entirely new state. This is exacerbated by the fact that the agent, block, and goal all initialize in a random location every episode. Because everything is changing every episode, it may have difficulty discerning which information is important and relevant to the successful series of actions, or trajectories.

CHAPTER 5

CONCLUSION

5.1 CONTRIBUTIONS

In this thesis, I clarified the distinction between collaboration, coordination, and cooperation and argued why collaboration is important to take robotic assembly to the next level. I presented an extensive review of multi-agent environments and assembly processes and evaluated existing coordinated interaction between robots. I presented an overview of reinforcement learning (RL), advances in Deep RL, as well as robotic control through reinforcement learning. I proposed a new theory of robotic collaboration and presented a framework for evaluating this research.

I designed a toy problem to enable the training of collaborative robots, and I developed and fabricated a team of multiple robots (two iterations). I built a new environment within the OpenAI gym framework wherein one or more agents to learn how to collaboratively move blocks–both centralized and decentralized control. I tested reinforcement learning algorithms (PPO, DDPG, MADDPG) for methods of collaborative assembly with multi-agent teams. I developed a framework for transferring learning from simulation to physical robots. Lastly, I compiled resources helpful for learning more about this research.

5.2 FUTURE WORK

As mentioned earlier, this thesis is only the first step in a larger effort to move towards autonomous collaborative robots. As such, there are many directions in which this research can and will go. One important step is to continue a focus on implementing this in the real world with physical robots. In this thesis, I developed a framework for transferring learning

from simulation to real-world. In continuing this direction, I would focus on system identification to better match the simulation to real world as well as make adjustments to both the physical robot and digital simulation. Another approach is to look at methods for speeding up learning in the physical world without the need for a simulation.

In addition, there are also many directions to go in terms of the different aspects of collaboration. In this thesis, I explored collaboration as a process whereby two agents with the same abilities learned to interact to push a block. Their abilities were somewhat limited in that they could only push and could not pull or grab. By working with another robot they were able to pinch, rotate, or guide the block together.

Another aspect of collaboration that I am interested in is when robots have different skillsets and abilities. Similar to when people come together to collaborate on a project, the robots would have no prior knowledge about the other's abilities and would need to communicate with each other and negotiate roles. The robots could have different sensing, strength or manipulation capabilities. They could also have varying knowledge of the goal or task at hand or intelligence and reasoning capabilities. Alone, neither robot would be able to complete the entire goal, but by working together and combining their skills they would be able to achieve much more. This is why there is power in collaboration. By creating a flexible system wherein robotic agents can adapt and learn over time, we can advance robotic assembly to operate in more dynamic and changing environments.

REFERENCES

Amato, C., Konidaris, G., Anders, A., Cruz, G., How, J. P., & Kaelbling, L. P. (2016). Policy search for multi-robot coordination under uncertainty. *The International Journal of Robotics Research*, *35*(14), 1760–1778. https://doi.org/10.1177/0278364916679611

Amato, C., Konidaris, G., Cruz, G., Maynor, C. A., Jonathan, P., & Kaelbling, L. P. (2015). Planning for Decentralized Control of Multiple Robots Under Uncertainty. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Institute of Electrical and Electronics Engineers (IEEE). Retrieved from https://pdfs.semanticscholar.org/d365/3502f61c56485bdbea7092e1652289e84e5b.pdf

Ann Marie Thomson, James L. Perry, & Theodore K. Miller. (2009). Conceptualizing and Measuring Collaboration. *Journal of Public Administration Research and Theory: J-PART*, (1), 23. https://doi.org/10.1093/jopart/mum036

Augugliaro, F., Lupashin, S., Hamer, M., Male, C., Hehn, M., Mueller, M. W., … D'Andrea, R. (2014). The Flight Assembled Architecture installation: Cooperative construction with flying machines. *IEEE Control Systems*, *34*(4), 46–64. https://doi.org/10.1109/MCS.2014.2320359

Ben-Ari, M., & Mondada, F. (2018). Robotic Motion and Odometry. In M. Ben-Ari & F. Mondada, *Elements of Robotics* (pp. 63–93). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-62533-1_5

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, *13*(Feb), 281–305.

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, *99*(suppl 3), 7280–7287. https://doi.org/10.1073/pnas.082080899

Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., … Vanhoucke, V. (2017). Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping. *ArXiv:1709.07857 [Cs]*. Retrieved from http://arxiv.org/abs/1709.07857

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *ArXiv:1606.01540 [Cs]*. Retrieved from http://arxiv.org/abs/1606.01540

Chen, L. (2012). Agent-based modeling in urban and architectural research: A brief literature review. *Frontiers of Architectural Research*, *1*(2), 166–177. https://doi.org/10.1016/j.foar.2012.03.003

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., … Zaremba, W. (2016). Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model. *ArXiv:1610.03518 [Cs]*. Retrieved from http://arxiv.org/abs/1610.03518

Colbry, S., Hurwitz, M., & Adair, R. (2014). Collaboration Theory. *Journal of Leadership Education*, *13*(4),

63–75. https://doi.org/10.12806/V13/I4/C8

Deisenroth, M. P. (2011). A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, *2*(1–2), 1–142. https://doi.org/10.1561/2300000021

Dogar, M., Knepper, R. A., Spielberg, A., Choi, C., Christensen, H. I., & Rus, D. (2015). Multi-scale assembly with robot teams. *The International Journal of Robotics Research*, *34*(13), 1645–1659. https://doi.org/10.1177/0278364915586606

Dreyfus, H. L. (1981). From micro-worlds to knowledge representation: AI at an impasse. In J. Haugel (Ed.), *Mind Design* (pp. 161--204). MIT Press.

Felbrich, B., Fruh, N., & Prado, M. (2017). Multi-Machine Fabrication. In *Disruptions + Disciplines*. Cambridge, Mass.

Gray, B., & Wood, D. J. (1991). Collaborative Alliances: Moving From Practice to Theory. *The Journal of Applied Behavioral Science*, *27*(1), 3–22.

Gray, B. (1989). *Collaborating: finding common ground for multiparty problems*. Jossey-Bass.

Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2016). Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates. *ArXiv:1610.00633 [Cs]*. Retrieved from http://arxiv.org/abs/1610.00633

Hall, E. L., & Hall, B. C. (1985). *Robotics: A User-Friendly Introduction*. New York: Holt Rinehart & Winston.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kassabian, P., Gumpertz, S., Menges, H. A., & Werfel, J. (n.d.). Towards Force-aware Robot Collectives for On-site Construction.

Knepper, R. A., Layton, T., Romanishin, J., & Rus, D. (2013). IkeaBot: An autonomous multi-robot coordinated furniture assembly system (pp. 855–862). IEEE. https://doi.org/10.1109/ICRA.2013.6630673

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274. https://doi.org/10.1177/0278364913495721

Levine, S., Finn, C., Darrell, T., & Abbeel, P. (2015). End-to-End Training of Deep Visuomotor Policies. *ArXiv:1504.00702 [Cs]*. Retrieved from http://arxiv.org/abs/1504.00702

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … Wierstra, D. (2015). Continuous control with deep reinforcement learning. *ArXiv:1509.02971 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1509.02971

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *ArXiv:1706.02275 [Cs]*. Retrieved from http://arxiv.org/abs/1706.02275

Macal, C. M., & North, M. J. (2005). Tutorial on agent-based modeling and simulation. In *Proceedings of the Winter Simulation Conference, 2005.* (pp. 14 pp.-). https://doi.org/10.1109/WSC.2005.1574234

Matarić, M. J. (1997). Reinforcement learning in the multi-robot domain. *Autonomous Robots*, *4*(1), 73–83.

Minsky, M. (2006). *The Emotion Machine: Commensense Thinking, Artificial Intelligence, and the Future of the Human Mind*. New York : Simon & Schuster, c2006.

Minsky, M., & Papert, S. (1971, December 11). Progress Report on Artificial Intelligence. MIT AI Laboratory. Retrieved from https://web.media.mit.edu/~minsky/papers/PR1971.html

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *ArXiv:1312.5602 [Cs]*. Retrieved from http://arxiv.org/abs/1312.5602

Mordatch, I., & Abbeel, P. (2017). Emergence of Grounded Compositional Language in Multi-Agent Populations. *ArXiv:1703.04908 [Cs]*. Retrieved from http://arxiv.org/abs/1703.04908

Navarro, I., & Matía, F. (2013). An Introduction to Swarm Robotics. *ISRN Robotics*, *2013*, 1–10. https://doi.org/10.5402/2013/608164

Ng, A. (n.d.). *Shaping and policy search in Reinforcement learning*. University of California, Berkeley. Retrieved from http://www.cs.ubc.ca/~nando/550-2006/handouts/andrew-ng.pdf

Niazi, M., & Hussain, A. (2011). Agent-based computing from multi-agent systems to agent-based models: a visual survey. *Scientometrics*, *89*(2), 479–499. https://doi.org/10.1007/s11192-011-0468-9

Petersen, K., Nagpal, R., & Werfel, J. (2011). Termes: An autonomous robotic system for three-dimensional collective construction. *Proc. Robotics: Science & Systems VII*.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., … Andrychowicz, M. (2017). Parameter Space Noise for Exploration. *ArXiv:1706.01905 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1706.01905

Rogers, D., & Whetten, D. A. (1982). *Inter-organizational Coordination: Theory, Research, and Implementation* (1st edition). Ames: The Iowa State University Press.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *ArXiv:1707.06347 [Cs]*. Retrieved from http://arxiv.org/abs/1707.06347

Sidor, S., & Schulman, J. (2017, May 24). OpenAI Baselines: DQN. Retrieved April 27, 2018, from https://blog.openai.com/openai-baselines-dqn/

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.

Stone, W. L. (2004). The History of Robotics. In T. R. Kurfess (Ed.), *Robotics and Automation Handbook*. CRC Press.

Sutton, R. S., & Barto, A. G. (2017). *Reinforcement Learning: An Introduction* (2nd edition, in progress). Cambridge, Massachusetts ; London, England: The MIT Press.

Testa, P. (2017). *Robot house : instrumentation, representation, fabrication*. New York, New York : Thames & Hudson, 2017.

Winograd, T. (1971, January). *Procedures as a Representation for Data in a Computer Program for understanding Natural Language*. Massachusetts Institute of Technology, Cambridge, Massachusetts.

Winston, P. H. (1970, January). *Learning Structural Descriptions from Examples*. Massachusetts Institute of Technology, Cambridge, Massachusetts.

Wood, D. J., & Gray, B. (1991). Toward a Comprehensive Theory of Collaboration. *The Journal of Applied Behavioral Science*, *27*(2), 139–162. https://doi.org/10.1177/0021886391272001

Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y., & Levine, S. (2016). Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search. *ArXiv:1610.00673 [Cs]*. Retrieved from http://arxiv.org/abs/1610.00673

FIGURES + TABLES

APPENDIX

LEARNING RESOURCES

Neural Networks and Deep Learning, by Michael Nielsen

- http://neuralnetworksanddeeplearning.com/index.html

CS231n Convolutional Neural Networks for Visual Recognition

- http://cs231n.github.io/convolutional-networks/

Andrej Karpathy's Pong from Pixels

- http://karpathy.github.io/2016/05/31/rl/

John Schulman's Lectures

- https://www.youtube.com/watch?v=aUrX-rP_ss4
- https://www.youtube.com/watch?v=8EcdaCk9KaQ&t=1653s

Deep Learning for Self-Driving Cars

- https://selfdrivingcars.mit.edu/

OpenAI Gym

- Documentation: https://gym.openai.com/docs/
- Github: https://github.com/openai

My multi-agent environments:

- https://github.com/khajash/multiagent-env