CARNEGIE MELLON UNIVERSITY

School of Architecture
College of Fine Arts

Thesis

Submitted in Partial fulfillment of the requirements for the degree of

Master of Science in Computational Design

TITLE:

Towards multi-drone autonomous construction via deep reinforcement learning

AUTHOR:

Zhihao Fang

ACCEPTED BY ADVISORY COMMITTEE

Prof. Daniel Cardoso Llach	Advisor	Date

Towards multi-drone autonomous construction via deep reinforcement learning

Zhihao Fang

ADVISOR:

Prof. Daniel Cardoso Llach

READERS:

Prof. Ramesh Krishnamurti
Prof. Josh Bard
Eddy Kim
Ardavan Bidgoli

CARNEGIE MELLON UNIVERSITY

Abstract

School of Architecture

Master of Science in Computational Design

Towards multi-drone autonomous construction via deep reinforcement learning

by Zhihao Fang

Autonomous construction has been an active research topic for engineers and designers for many years. Meanwhile, technological advancement made in the drone industry is continuously pushing the drone's capability boundary. The probability of drones actively participating in additive construction is large enough to be realized in the near future. However, there is no system that can control a scalable number of drones for autonomous construction in a dynamic continuous environment. This thesis aims to develop a system for autonomous multi-drone additive construction using deep reinforcement learning-based algorithm. First, the process of multi-drone additive construction is modeled in a computer simulation. Then state-of-art deep reinforcement learning algorithm is applied to achieve collision avoidance in navigation. A software package is then developed and able to be integrated into a 3d modeling software, Rhinoceros, for future use and development for researchers and designers. Finally, this system is applied in two experiments: bricklaying and facade coating to demonstrate usage.

Acknowledgements

I wish to express my sincere appreciation to the following people.

My supervisor, Prof. Daniel Cardoso Llach, who has supported my research from the very beginning. He convincingly guided me to be professional at doing research and gave me inspirations and advices throughout the research.

My thesis readers, Prof. Josh Bard, Prof. Ramesh Krishnamurti, Eddy Kim and Ardavan Bidgoli for their precious comments on my thesis.

My family and friends, who supported my study in the past two years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents a system for controlling a team of drones for additive construction. It models the multi-drone autonomous construction problem, develops a system to solve the problem and finally develops a 3D modeling software integration for future use by researchers and designers.

1.1 Prologue

The emergent of new design and construction technologies could potentially lead to a new paradigm shift in architecture. In recent decades, parametric software, robotic arms, 3D printing open up new fields in architectural research. These fields, also known as parametric design and fabrication, coupled designers with the process of materialization, provide designers the power to realize more innovations.

In recent years the technology advancement and popularization of drones with the back up of sensing technology and ML algorithms makes drones become a new potential tool in architecture design and construction. However, there is still a huge gap identified in existing works. The first attempt of drone-based construction can be dated back to 2011, when researchers from UPenn used teams of quadcopters to assemble a 2.5D truss-like structure (Lindsey, Mellinger, and Kumar, 2012). In the same year, the *Flight Assembled Architecture* by Federico et al. (Augugliaro et al., 2014) shows the potential of aerial construction by installing a 6-meter tall tower consisting of 1500 foam elements by four quadcopters. Compared to ground robots, drones are more agile and efficient if work in teams, they are less restricted by workspace constraints. However given the fact aerial construction is almost a ten-year-old research field, there are not many published works and problems like scalability and extensibility are identified in existing works. Apart from hardware limitations like drone load capacity or precision, one critical reason is there is either no framework that models the multi-drone autonomous construction problem or system that solves the problem. Recent years, under the boom in artificial intelligence, deep learning and reinforcement learning are being adopted in robotics to achieve intelligence in robot's decision making. Algorithms are developed to solve multi-robot problems, however, there is still no application regarding multi-drone construction.

Among the most difficult parts of multi-drone autonomous construction problem is multi-drone collision-free coordination. Base on state-of-art reinforcement learning algorithm, this thesis tries to model and solve the problem.

The ultimate aim of this thesis is to make designers experiment more easily in aerial construction, to open up new possibilities for drone research, to pave way for a new paradigm shift in architecture design and construction.

1.2 Statement of problem

Existing works on multi-drone autonomous construction have limitations of scalability and extensibility. While great advancement is made in reinforcement learning, there is no application in the field of multi-robot autonomous construction. This thesis models the problem of multi-drone autonomous construction and develops a system based on deep reinforcement learning that is:

- applicable to real-time multi-drone autonomous construction scenarios
- scales well when the number of drones and the size of the structure grows up
- collision avoidance in dynamic environments
- supports integration with an architecture modeling software

1.3 Outline of chapters

- Chapter 2 explores the technical background of drone and existing works on drone-based construction to show the potential of drones actively participate in the construction. Background of reinforcement learning, together with related works are then visited in order to formulate the problem under an RL framework.
- Chapter 3 models the multi-drone autonomous construction problem. Based on that, the system is developed, system architecture, RL model, and 3D modeling software integration are described.
- Chapter 4 first describes the training process of the RL model. Then two simulation experiments: bricklaying and facade coating, are developed on the system, and results are shown.
- Chapter 5 makes a conclusion, clarifies contribution and limitations, and suggests future works.

Chapter 2

Background

In recent decades, researchers have been trying to use robots to automate parts of construction progress. The combined innovation in both architecture design and engineering can lead to another paradigm shift in architecture. Recent advancements made in drone technology and application open up possibilities for drone construction. This chapter introduces three important background of the thesis: drone technology, drone-based construction, and reinforcement learning. Based on these backgrounds, this thesis could model and solve the multi-drone autonomous construction problem.

2.1 Drone

Unmanned aerial vehicle(UAV), also know as drone, is much smaller and more agile compared to ground robots. The flight ability makes its workspace have almost no constraints. It can be transported to construction site easily, be deployed in all kinds of workspace, work in a fleet to gain efficiency, and reach places which are too high for ground robots to reach. Although there is various kinds of drones, this thesis refers drones to the typical multirotor drones.

2.1.1 Hardware

A drone typically consists of a flight controller, a frame, propellers, motors, a battery, sensors, and transmitters (Fig. 2.1). A radio controller or a ground station can control the drone.

The flight controller is a circuit board that interprets input from a receiver or other onboard sensors to control the drone. One of its most important components is inertial measurement unit(IMU) which works by detecting acceleration to estimate drone's pitch, yaw and roll to achieve stabilized flight. The flight controller utilizes a PID control loop in software to maintain stability during flight.

Propellers and motors mounted on the drone frame make drones fly. Drones can be classified by the number of motors. Common multirotor drones can be categorized into quadcopter, tricopter, hexacopter, and octocopter. The selection of motors directly impacts drone size and capability.

Battery management of a drone is critical in application cases since the battery capacity of a drone is quite limited. Studies (Park, Zhang, and Chakraborty, 2017; Shin, Kim, and Levorato, 2019) have been conducted on battery management for a fleet of drones, showing the importance of battery management on large scale, long lasting tasks.

FIGURE 2.1: Hexacopter platform hardware configuration and components assembly (Braga, Silva, and Ramos, 2016)

Methods of localization include GPS sensor, allows precision within meters, post-processed kinematic (PPK), Real Time Kinematic (RTK) system, or Ultra-Wideband localization system(UWB) brings centimeter precision(Ammar, 2016). Apart from measuring position externally, drones can also use surrounding images taken by cameras to perform localization, one representing algorithm is SLAM (Bryson and Sukkarieh, 2007), which simultaneously updates a map of the environment. However, vision-based localization requires heavy computation power for onboard computers and the precision is usually not high. A commonly adopted way in researches to localize a drone is to use a motion capture system. By installing motion capture markers on-board and using external sensors, one could achieve millimeter precision. However, motion capture usually requires an indoor environment.

Additional devices mounted on the drone gives it ability other than flying. For the most cases, cameras are installed on commercial drones to capture photo and video. End effectors like grippers need to be specially designed to be installed, considering size, load capacity and power of drone.

2.1.2 Control Systems

Different sensors are connected to a flight control unit (FCU) (e.g. Pixhawk) to gain a low-level control of drone's motors. The FCU could receive data from a radio controller for direct control or from an onboard computer (e.g. Raspberry Pi) to gain more customizable control on drone's behaviors including end effectors. MAVLink is a messaging protocol for communicating with drones or within their onboard components. Additionally, the on-board computer can receive data from a remote ground station computer, to make the drone more programmable. The ground station computer can monitor states of the drone through communication between the on-board computer, it could also transmit data or instructions, for instance, external motion capture data, into the on-board computer. Besides, path planning softwares can run on the ground station. Robot Operating System (ROS) is a also a have-to-mention widely used software

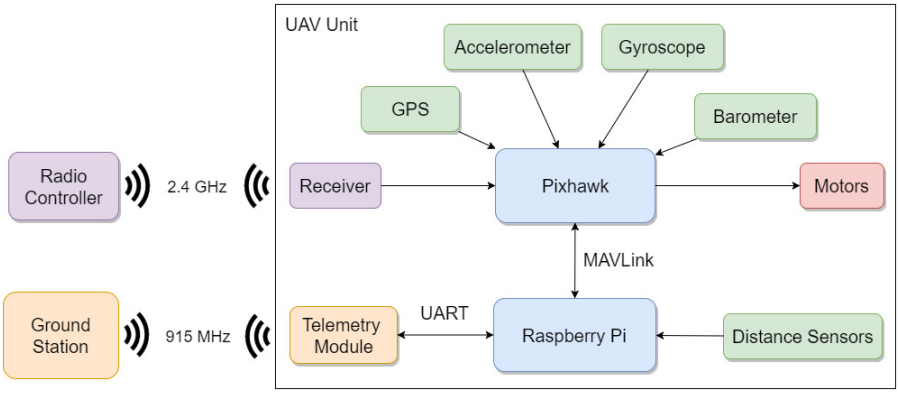


FIGURE 2.2: UAV control system (Braga, Silva, and Ramos, 2016)

system for ground station, which provides ROS bridge for communication, and other open-source packages for development and software simulation.

2.1.3 Civilian Drone Applications

Although the drone is originated in military applications, the prevalence of civilian drones now surpassed military drones. Technology advancement makes drones have a wide application includes policing and surveillance, product deliveries, photography and filming, inspections, agriculture, drone display, and drone racing(Wikipedia contributors, 2020b). The existing application in construction sites is mainly inspection, such as site planning, progress monitoring, photogrammetry based 3D reconstruction(Kaamin et al., 2018; Coetzee, 2018).

2.1.4 Drone swarming

Controlling a fleet of drones to achieve swarm behavior requires robust path planning. Drone swarming technology is almost exclusively used in drone display, which flies drones in a synchronous coordinated fashion for public display. Drone display can be dated back to 2012 (Wikipedia contributors, 2020a), and the record for the most drones simultaneously controlled by one pilot is 2066 drones by Intel Shooting Star[1]. Equipped with LED, drone swarms entertain the audience by forming various arrays in 3D or performing swarming behaviors. To be noticed, whether involving swarming behaviors or not, trajectories for drone display are pre-designed by planning algorithms since environment uncertainty is small. The drone display performs in clear sky and the state of every drone at any time stamp is certain.

2.2 Drone-based Construction

With the development of drone technology, one can foresee the future of drones actively participating in the construction. Flying in midair, drones have fewer constraints on

[1] https://www.intel.com/content/www/us/en/technology-innovation/drone-light-show-fact-sheet.html

workspace than ground robots and larger reachable area. Willmann et al. (2012, p.442) listed advantages of Aerial Robot Construction (ARC) over standard robotic systems:

> "First, because aerial robots fly and mount construction parts directly to their required position, ARC does not require scaffolding and is less constrained by height and bottom-up accessibility. Second, ARC structures can be built according to highly complex designs: aerial robots operate under the explicit guidance of digital architectural design, and can place and manipulate material according to a precise digital blueprint. Third, ARC work capacity is easily scalable: while conventional machines are limited to operating on a small component of a traditional structure, many aerial robots can operate on an ARC structure at the same time, either individually or cooperatively."

2.2.1 Autonomous Construction

Although robotic arms are widely experimented to achieve construction tasks, they are used in the off-site prefabrication, components being fabricated usually need to be transported to the site. Demolish robots are also developed for commercial use, they require operators. Until recently, some ground robots have been developed for autonomous in-situ construction (Figure 2.3). Ground robots have the advantage of load capacity and precision. However, size, mobility, and reachability limit their capabilities which lead to difficulties of transporting to the construction site, navigating in the site, and restricted construction range.

This thesis is based on additive construction, which can be categorized, in terms of method and material, into discrete material construction and continuous material construction. For discrete material construction, materials are usually block-like, the target location and orientation are the only concerns for such types of construction. Discrete materials can be deployed instantly. For continuous material construction, materials are deployed in a continuous time and form, like 3D printing materials or tensile materials. Apart from target location and orientation, the path drones follow in deploy time also matters. It is inappropriate to automate material deploying process using stochastic algorithms except to achieve some intended randomness.

In a multi-robot collaboration scenario, additive construction can be categorized into synchronous collaboration and asynchronous collaboration. In a synchronous situation, robots need to coordinate together to perform a single task. E.g. two drones need to coordinate to weave a spacial tensile knot. In an asynchronous situation, robots can perform a single task without help from others, e.g. bricklaying. Discrete construction usually falls into asynchronous collaboration and continuous construction may fall into both categories.

Multi-drone Collaboration

The advantages of multi-drone collaboration are quite obvious: improved efficiency and ability, to achieve difficult tasks. However, the cost of deploying multiple drones does not grow linearly with the number of drones since hardware and software need to be robust to support multiple drones.

Multi-drone collaborative construction methodology is different from drone display (mentioned in 2.1.4). The task of a drone in drone display is to move to a number of

(A) bricklaying robot SAM100 (internet source) (B) In-situ Fabricator (Giftthaler et al., 2017)

FIGURE 2.3: Different ground robots for construction automation

points following pre-planned trajectories under limited environment uncertainties. Although pre-planning trajectories for drone construction might be possible for a known environment and small numbers of drones and tasks, one solution may not be applicable to a slightly different scenario. Moreover, things quickly get much more complicated and unpredictable when raising the number of drones or tasks. Multi-drone construction's environment which this thesis tries to model is uncertain for pre-planning. What's more, the way tasks are assigned and executed introduces randomness into the system. In conclusion, the system for drone display is not applicable to multi-drone autonomous construction.

2.2.2 Case Study

Below are several published works using drones in architecture fabrication or construction. Each of them made a great contribution towards aerial construction and showed a promising future of the application of drones in construction. However, the efficiency advantage of drone teams is not made enough use of. A single drone or only a few drones are used in these cases, a complexity gap can be identified when the number of participating drones grows up and to the best of the author's knowledge, there is no system developed for multi-drone autonomous construction.

Construction of Cubic Structures with Quadrotor Teams (Fig. 2.4)

In 2011, Quentin et al. from University of Pennsylvania developed a system for teams of quadcopters assemble 2.5D truss-like structures (Lindsey, Mellinger, and Kumar, 2012). The authors designed the truss-like structure consisting of "beams" and "columns" with magnets embedded at joints, along with a gripper, which can pick up elements laid horizontally and vertically. They then introduced an algorithm for a team of drones taking turns to pick up elements from supply station, transport to the construction area, and finally assemble at designated positions following a fixed route.

The Flight Assembled Architecture Installation (Fig. 2.5)

Federico et al. built a 6-m tall tower composed of 1500 foam modules by four quadcopters during a live exhibition at the Fonds Régional d'Art Contemporain du Centre

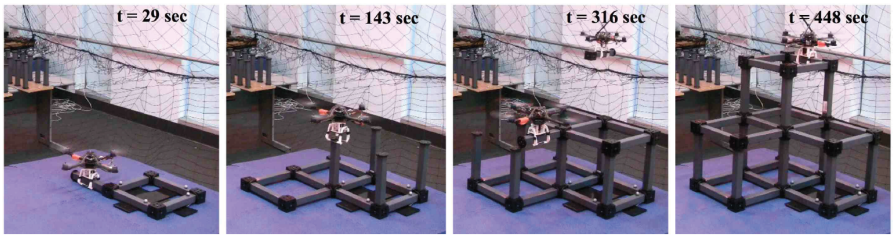


FIGURE 2.4: Snapshots of a special cubic structure being built by three quadrotors (Lindsey, Mellinger, and Kumar, 2012)

in Orléans, France in 2011 (Augugliaro et al., 2014). This project showed a whole process for additive drone-based construction. Four drones worked collaboratively, each performed whole process of picking up a foam brick, transporting to build area, placing the brick and charging. Gripper was specially designed to be robust for picking and releasing bricks. Though the project was built on the ETH Flying Machine Arena (FMA) platform, which was a research platform for controlling fleets of micro drones at a low level, placing trajectory was still studied thoroughly for high precision (25 mm).

For trajectory planning, mainly regarding moving between stations and the build area, this project used a planner consisted of three subsystems. First, allowable fly regions defining, then waypoint-based navigation coupled with space reservation system enables the discretization of flyable space, finally a trajectory planner was used to generation trajectory from point to point (Augugliaro et al., 2014). Although the space reservation mechanism allowed drones to fly without collision, it might cause deadlocks when two drones try to swap positions. This problem was solved by defining two free-way at different heights, each of which can only be accessed by one drone at any time. However, using of free-way requires predefining, which might be difficult for sophisticated structures or dynamic environments. Additionally, it limits the number of drones flying simultaneously.

FIGURE 2.5: The Flight Assembled Architecture installation (Augugliaro et al., 2014)

3D Printing with Flying Robots (Fig. 2.6)

This is the first of few attempts trying to integrate 3d printing module in a quadcopter (Hunt et al., 2014). The paper went into depth in printing material choosing, printing system development, and integration. Experiments on different scenarios are performed and results showed aerial 3D printing was a promising field. Finally, this paper pointed out that multi-drone coordination enables the construction of larger structures as future work.

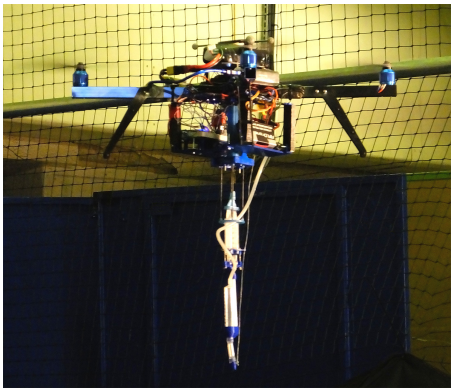


FIGURE 2.6: Aerial 3D printer, quadcopter with integrated printing module in orientation ready for printing (Hunt et al., 2014)

Aerial Construction for Tensile Structures (Fig. 2.7)

Ammar Mirjan has done a series of aerial construction experiments with innovative tensile structures (Ammar, 2016). The most developed one is a full scale, load-bearing footbridge, spanning 7.4 m. The series of work was also built on the same FMA system as The Flight Assembled Architecture Installation. Two drones synchronically collaborated to weave ropes according to designed trajectories. Since trajectories determined the structure form, they were crafted in the digital design tools. By using a tensile structure as an innovative form, this work utilized drone's characteristic of agility and collaboration, the result of which was hard for other robots to achieve.

Cyber physical macro material as a UAV [re]configurable architectural system (Fig. 2.8)

The research by Wood et al., 2018 describes a new strategy for deploying a cyber-physical macro material combined with aerial construction robots as a reconfigurable architecture. A reconfigurable canopy system consisted of a changeable number of material units was designed to provide intelligent dynamic urban space for the public. Instead of picking up units from the supply station, the drone directly picked up units attached on the canopy and attached it to another position intelligently controlled by a central system.

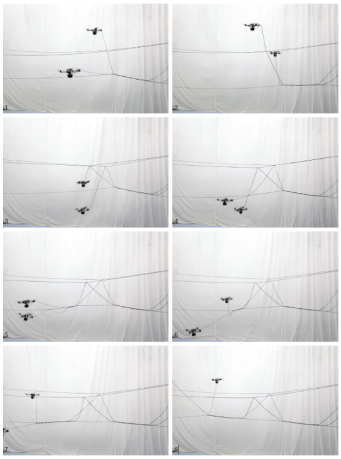


FIGURE 2.7: Two quadcopters building part of a bridge
(Ammar, 2016)

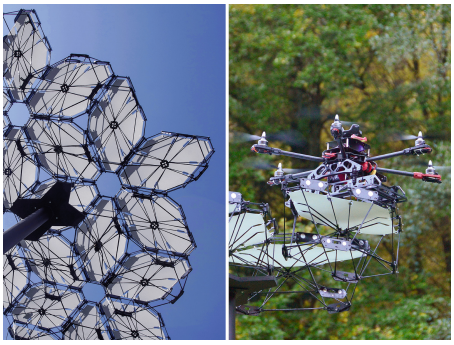


FIGURE 2.8: Quadcopter building part of a canopy
(Wood et al., 2018)

Facade coating by drones(Fig. 2.9)

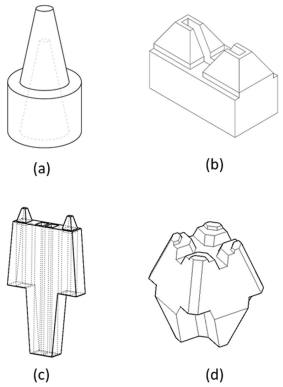
MuDD Architects is experimenting with drone spraying for facade coating in a series of projects. MuDD Architects offers solutions for quickly building sustainable architecture using local natural materials. In the project of Mud Shell, a shelter made from bags of hay attached to a wooden lattice. Then a drone was used to coat a mixture of clay and fibre. Although the work requires workers to manually control the drone, which is not exactly in line with the autonomous construction approach, the innovative usage of drone in coating suggests a new possibility in the drone-based construction field.

2.2.3 Feasibility Study and Limitations

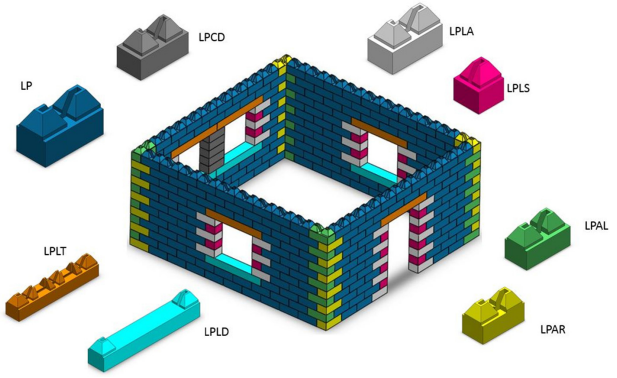
Although research works in drone-based construction seem to be promising, the gap is still identified between research and real-world application. Some pioneer studies (Latteur et al., 2015; Goessens, Mueller, and Latteur, 2018) regarding the feasibility of deploying drones on real construction cases. Drones precision, behavior while transporting, handling, and laying loads as well as drone compatible construction elements are investigated.

FIGURE 2.9: The Mud Shell project
(image from MuDD Architects)

Weight and tolerance are taken into account when Goessens, Mueller, and Latteur, 2018 tried to design drone compatible elements (2.10a) and discretize architectural structure into these elements (2.10b). The experiment also showed that a 12kg drone can lift and lay concrete blocks above 20kg controllably. A laying tolerance of 5cm had to be considered in the design of the geometry of blocks. The laboratory test showed that the construction time of a drone-based masonry wall could lead to a cost up to 10 times cheaper, due to one drone being up to 5 times faster than a worker.



(A) four families of dricks



(B) Eight different dricks are needed to cover every straight wall configuration

FIGURE 2.10: Drone compatible elements
(Goessens, Mueller, and Latteur, 2018)

2.3 Reinforcement Learning

Significant advances have been made in reinforcement learning (RL) in recent years. RL algorithms learn to make decisions in many problems including Go, robotic control, autonomous driving, game AI, etc. To the best of author's knowledge, using deep RL to achieve multi-drone autonomous construction is still an unexplored field. Although state-of-art RL algorithms (Fig. 2.11) have exhibited huge success in empirical success, they make different assumptions about the environment. In order to model a multi-drone construction problem and choose the algorithm properly, this section reviews the basic RL knowledge framework by scratching the surface of RL. First, single-agent RL

and multi-agent RL are visited for problem setup and algorithm sort out. Second, a few works which utilize RL on the multi-drone application are reviewed. Then the way that the RL algorithm is applied to the multi-drone autonomous construction problem is discussed. Finally, a related work, which this thesis based on, is reviewed. The MARL overview paper by Zhang, Yang, and Başar, 2019 is referenced for this section.

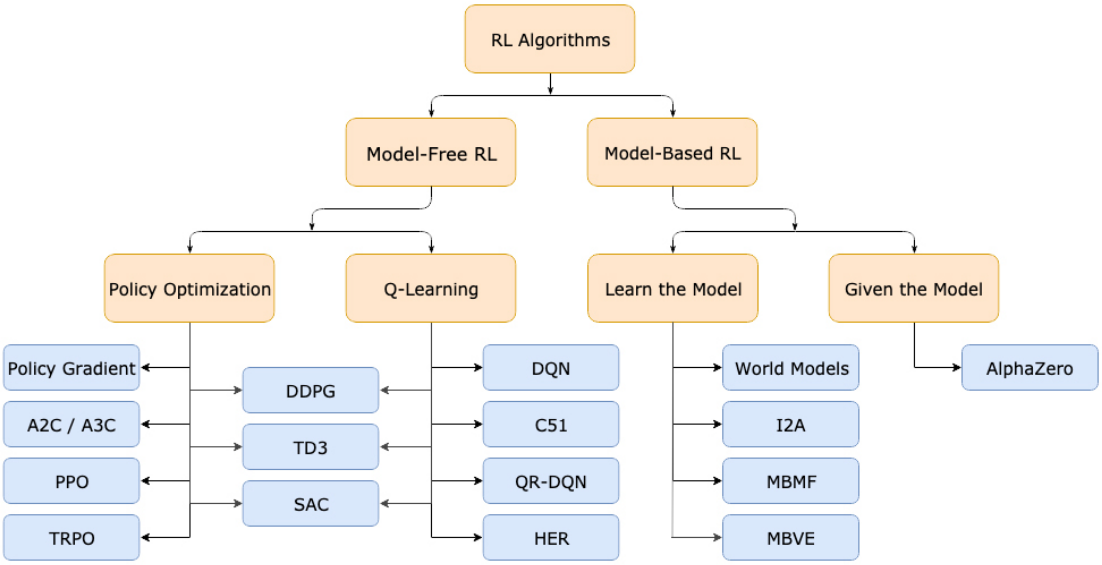


FIGURE 2.11: A non-exhaustive, but useful taxonomy of algorithms in modern RL (OpenAI)

2.3.1 Terminology

Some basic terminology for RL used in this thesis is listed, inspired by class slides of Deep Reinforcement Learning and Control course at Carnegie Mellon University.

Agent - An entity equipped with sensors and end effectors to interact with the environment. In this thesis, an agent refers to a drone.

Observation - The input of agent's sensors.

State - A state, usually associated with a timestamp, is the available information of an agent about the environment.

Action - Action is what the agent acts to interact with the environment.

Reward - A scalar the environment gives to the agent each time after the interaction to indicate whether the agent achieved the goal or not.

Return - Cumulative sum of (potentially time discounted) rewards. Maximizing expected return is basically the learning goal of RL algorithms.

Model - The mapping function from state/observations and actions to future states/observations.

Model-based/Model-free RL - Whether dynamics are known/estimated in the learning.

Policy - The mapping function from states to actions of the agent. In this thesis, the policy is approximated by a deep neural network.

On-policy/Off-policy - On-policy methods update the policy that is used to generate data, while off-policy methods update a policy differently.

2.3.2 Single-agent RL

In a single-agent RL setting, the agent learned to perform decision-making by interacting with the surrounding environment, which is usually mathematically modeled as a Markov Decision Process (MDP), defined by state and action spaces, a transition function, a reward function, and a discount factor. In MDP the agent has access to an observation of the environment. The goal of solving the MDP is to find a policy that maximizes the discounted cumulative reward:

$$\mathbb{E}[\sum_{t\geq 0} \gamma^t R(s_t, a_t, s_{t+1}) | a_t \sim \pi(\cdot|s_t)]$$

When the environment model is fully known, Dynamic Programming can iteratively evaluate value functions and improve policy. However, it is often impossible to gain perfect knowledge of the model. There are two main types of RL algorithms by large, value-based methods and policy-based methods. Value-based methods aim to learn an optimal value function while policy-based methods aim to directly optimize the policy.

Value-based methods

Q-learning is a typical model-free off-policy value-based algorithm. The agent updates its state-action value function (Q function) after making a step in the environment, then the approximate optimal policy can be obtained by taking the greedy action of the Q function. Deep Q-learning is a breakthrough deep RL algorithm where Q function is approximated by a neural network. Other value-based methods include TD learning, SARSA, Monte-Carlo method, etc.

Policy-based methods

In policy-based methods, the policy is estimated by parameterized function approximators (e.g. neural networks). Based on the Policy Gradient therom, the parameter of the policy function can be updated. By estimating the gradient in various ways, policy-based algorithm family includes: Actor-Critic algorithms (Konda and Tsitsiklis, 2000; Bhatnagar et al., 2007), Deterministic Policy Gradient algorithms (Silver et al., 2014), Trust Region Policy Optimization (Schulman, Levine, and Abbeel, 2015), Proximal Policy Optimization (Schulman et al., 2017), Soft Actor-Critic (Haarnoja et al., 2018), etc. Since using neural networks for function approximation, policy-based methods can handle more complex action space than value-based methods.

2.3.3 Multi-agent RL

Multi-agent RL (MARL) extends single-agent RL by introducing multiple decision making agents. In particular, the state and reward received by each agent are influenced by the joint actions of all agents. A generalization of MDP of multiple agents is Markov games, defined by a number of agents, state space, joint action space, transition function, joint reward function, and discount factor. Similar to single-agent RL, each agent tries to optimize its long-term reward. Various Multi-agent RL settings include the cooperative setting, where all agents share a common reward function, the competitive setting, where competing agents exist and the sum of rewards is zero, and the mixed

setting, where there is no restriction on goal and agents relationship. However, both the Markov decision process and Markov games can only model agents have a full observation on the environment, while most cases agents are restricted to have only partial observability. That formulated the Partially Observed MDP (POMDP). Lots of works assume the existence of a central controller that has knowledge of joint actions, joint rewards, and joint observations, which gives birth to the widely adopted *centralized-learning-decentralized-execution* paradigm.

According to the review of Zhang, Yang, and Başar, 2019, there are several challenges that MARL algorithms are commonly facing across different settings. First, the learning goals in MARL are multidimensional, that brings up the challenge of dealing with equilibrium points. Second, for each agent, the environment becomes non-stationary since all agents are updating their policies according to their own interests concurrently. Finally, the joint action space increases exponentially with the number of agents may cause scalability issues.

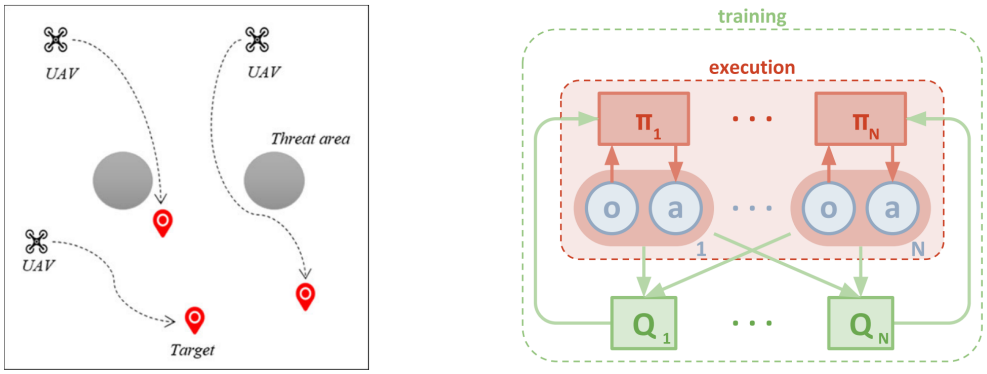
Current state-of-art MARL algorithm frameworks include: MADDPG (Lowe et al., 2017), QMIX (Rashid et al., 2018), VDN (Sunehag et al., 2017), COMA (Foerster et al., 2017), QTRAN (Son et al., 2019), CommNet (Sukhbaatar, Szlam, and Fergus, 2016), DyMA-CL (Wang et al., 2019), G2ANet (Liu et al., 2019), Mean Field RL (Yang et al., 2018; Carmona, Laurière, and Tan, 2019), etc.

2.3.4 MARL based drone applications

There are several works using MARL to control a team of drones for cooperation tasks in a decentralized fashion. Yang and Liu, 2018 addresses the drones' optimal links discovery and selection problem in uncertain environments for information sharing. Pham et al., 2018 proposes a distributed MARL algorithm for a team of drones to learn cooperatively to provide full coverage of an unknown field of interest while minimizing overlapping. Shamsoshoara et al., 2018 develops a distributed mechanism for spectrum sharing among a network of drones and licensed terrestrial networks. The drones learn the optimal task allocation using a distributed RL algorithm. Tožička et al., 2019 develops a control system for controlling a fleet of drones using a centralized CNN policy optimized by Double Q-learning. The policy network output a density map for drone distribution by finding modes of the map. This approach might be problematic since the mapping of drones to the modes is not articulated. Besides, this work discretized the environment into a grid world, which restricted drones' flying area and action space. Qie et al., 2019 considered the problem of multi-drone simultaneous target assignment and path planning (STAPP) (Fig. 2.12a). Although the work shares some similarities with this thesis, the authors assume full access to threats and other drones at execution time, which doesn't fit the problem modeled in this thesis (3.1). What's more, the MADDPG framework (Lowe et al., 2017) approach they use suffers scalability problem since centralized critic cannot deal with the varying number of agents. (Fig. 2.12b).

2.3.5 Multi-drone navigation problem

Construction is a complex process where automation can engage in different phases such as navigation, resource allocation, intelligent construction, and collaboration. Since there is no existing work on the multi-drone construction system, this thesis aims to be



(A) Multi-UAV targets assignment scenario (Qie et al., 2019)

(B) MADDPG framework (Lowe et al., 2017)

FIGURE 2.12: STAPP and its framework

generic at problem modeling to allow extensibility for various scenarios. With more details discussed at 3.1, this thesis uses an RL algorithm to solve one of the most critical aspects of the problem: multi-drone collision-free navigation in a dynamic environment under partial observability of agents. Two main features of the algorithm are expected: scalability and continuous control. Moreover, the algorithm should be able to generalize to the scenarios with dynamic obstacles and other proactive agents.

Non-RL collision avoidance

Collision avoidance is a long-studied topic. According to Long et al., 2017, typical *non-RL* solutions can be classified into centralized methods and decentralized methods. Centralized methods do path-planning (such as A^* algorithm) for all agents simultaneously, which require knowledge about the environment and have poor scalability: all computation is done on the server concurrently. What's more centralized methods rely heavily on the communication between the server and agents. An example is made by Honig et al., 2018 in which the authors achieved path planning for a swarm of drones.

On the other hand, decentralized methods in an agent-level setting, where agents independently make decisions based on the estimation of the state of other agents or environment entities. Velocity obstacle (VO) algorithm and its variants (such as ORCA by Berg et al., 2011, NH-ORCA by Alonso-Mora et al., 2013) are widely used agent-level decentralized collision avoidance algorithms, however, this kind of method requires agent to have a high precision agent-level estimation of the environment, which heavily depends on sensor's performance. It is also reported the performance of navigation speed and time is much lower than their centralized counterpart. Besides, these algorithms have many parameters that are difficult to be tuned properly. (Long et al., 2017).

Scalability

Scalability is one of the most important factors need to be considered for developing the multi-drone construction system since the number of drones at deploy time is unknown, which might also be large. Under the context of multi-drone autonomous construction, the system is expected to comprise drones from a few to dozens, or even more. Methods

Algorithm	Policy	Action space	Operator
A3C	On-policy	discrete/continuous	Advantage
A2C	On-policy	discrete/continuous	Advantage
ACER	Off-policy	discrete/continuous	Advantage
ACTKR	On-policy	discrete/continuous	Advantage
SAC	Off-policy	continuous	Advantage
TRPO	On-policy	discrete/continuous	Advantage
PPO	On-policy	discrete/continuous	Advantage
NAF	Off-policy	continuous	Advantage
DDPG	Off-policy	continuous	Q-Value
TD3	Off-policy	continuous	Q-value

TABLE 2.1: Deep RL algorithms for continuous control

that only support a small number of agents, or require the same number of agents at training and execution time, are not qualified for the scalability requirement. Instead of agent-level observation, sensory input for observation is preferred since the input dimension is fixed, it will not suffer from dimension problems. What's more, under the assumption of homogeneous exchangeable agents, agents can learn independently, so that some single-agent RL algorithms becomes applicable even under the multi-agent setting.

Continuous control

This thesis aims to be applied to continuous action and space settings. Robot control in the physical world is usually continuous, not to mention drones, which is eventually the control of motor power. Discretizing action space hinders the robot's actual ability and precision. The continuous action space also implies a continuous environment state space. Discretizing environment (e.g. into the grid world) requires pre-processing the environment, which breaks the assumption of dynamic environment application. Flight area and precision are also potentially restricted for environment discretization. Table 2.1 listed state-of-art single-RL algorithms for continuous control, which all could potentially be applied to a multi-drone navigation problem.

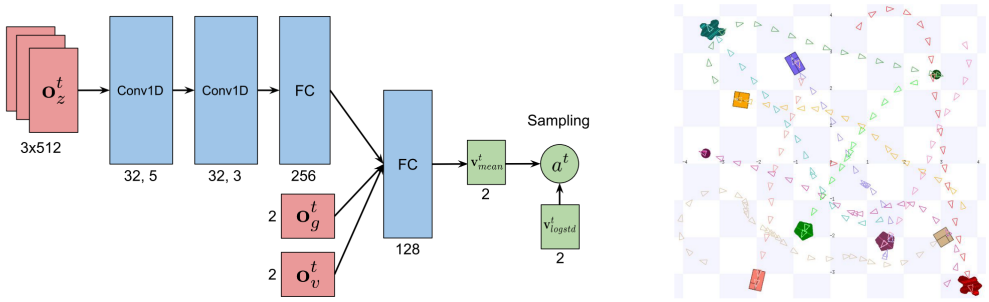
2.3.6 Related Work

For the RL part, this thesis adapted methods from the following paper with modification: *Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning* by Long et al., 2017. Although there exist other works achieving similar or even better results, it is difficult to do an exclusive search on the topic of robot navigation, where there is a vast amount of research and the performance evaluation standard is various. The selection of this paper is based on its environment settings and promising empirical performance.

The authors present a decentralized sensor-level collision avoidance policy for multi-robot systems, which directly maps raw sensor measurements to an agent's steering commands in terms of movement velocity. In the paper's setting, robots are modeled as discs on the Euclidean plane with obstacles and other decision-making robots. Robots

operate in a fully decentralized manner and are given a partial observation of the whole environment. Observation space consists of a 2D laser range finder sensory input (consists of 512 laser distances $\times$ 3 time steps), a 2D vector indicating relative position between the robot and its goal in polar angle, and a 2D vector representing the current translational and rotational velocity. The action space is a 2D vector including translational and rotational velocity. An agent is rewarded by arriving at its goal of making an approaching step, penalized by getting away from its goal, collision, or large rotational velocities.

The neural network architecture is given in Fig. 2.13a, the final action is sampled from the Gaussian distribution constructed by v^t_{mean} with a separated log standard deviation vector v^t_{logstd}. The authors extended PPO to a multi-robot setting to train the model. What's more, in order to train robots in diverse environments, the authors used multi-scenario and multi-stage to achieve curriculum learning. In the evaluation section, the trained model is used to compare with a state-of-art policy, NH-ORCA, trained by supervised learning and showed a large improvement in performance.



(A) Policy network architecture (B) Result

FIGURE 2.13: *Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning* (Long et al., 2017)

Chapter 3

Method

In this chapter, methodology including problem modeling, system design, and RL algorithm will be visited. Based on the way the problem is modeled, a simulation-based system is designed and built to solve the multi-drone autonomous construction problem. As a key component of the system, the RL algorithm part will be carefully reviewed and there is another layer of abstraction from the simulator to what the RL algorithm solves. This chapter discusses system architecture, algorithms, and integration as a general framework. In order to use the system in specific scenarios, extra implementations should be done accordingly, which is introduced in Chapter 4.

3.1 Problem Formulation

Multi-drone autonomous construction involves several levels of complexity: hardware, control, planning, design, management, etc. Considering the complexity and different application scenarios, it is hard to build an end-to-end system to solve all the problems. However, by properly modeling the problem, the problem can be decomposed and solved part by part so that some parts that are applicable in all scenarios, can be directly implemented by the framework, while other parts can be implemented by users according to specific cases. In the multi-drone autonomous construction problem, coordinating an uncertain number of drones without collision in a dynamic environment is one of the most difficult parts, which this thesis focuses to solve.

3.1.1 Overview

This thesis mainly focuses on general problem modeling and solving algorithm part of the multi-drone autonomous construction problem, which is built in a software simulator environment, where the environment and drone's control is modeled to be continuous while the time is discretized.

For a high-level intuition, the system coordinates multiple drones to perform additive construction. Illustrated in Fig. 3.1. Initialized at a charge station, a drone needs to resupply construction material at the supply station and deploy it at the designated position. During navigation, it should avoid other drones and possible threat area. The drone needs to charge at the charge station when the battery is low. Drones, building material, supply stations, charge stations, threat areas are defined as entities. Drones are treated as independent agents, which communicate with the server in an asynchronous manner. Scalability and extensibility are two key aspects the system tries to feature.

As stated above, some part of the problem is applicable across different use cases, such as main construction loop, entities' functionality and basic behavior, and navigation algorithm. These subproblems should be modeled and implemented in the system. While other subproblems could differ case by case, they are modeled at a minimum viable manner and assumed to work in this thesis's implementation. Users need to elaborate implementation of these parts depending on the use case. Moreover, this thesis assumes drones have enough load capacity and precision to successfully perform tasks because addressing these problems relies on hardware implementations which are not concerned. Users should address these problems when implementing drone tasks.

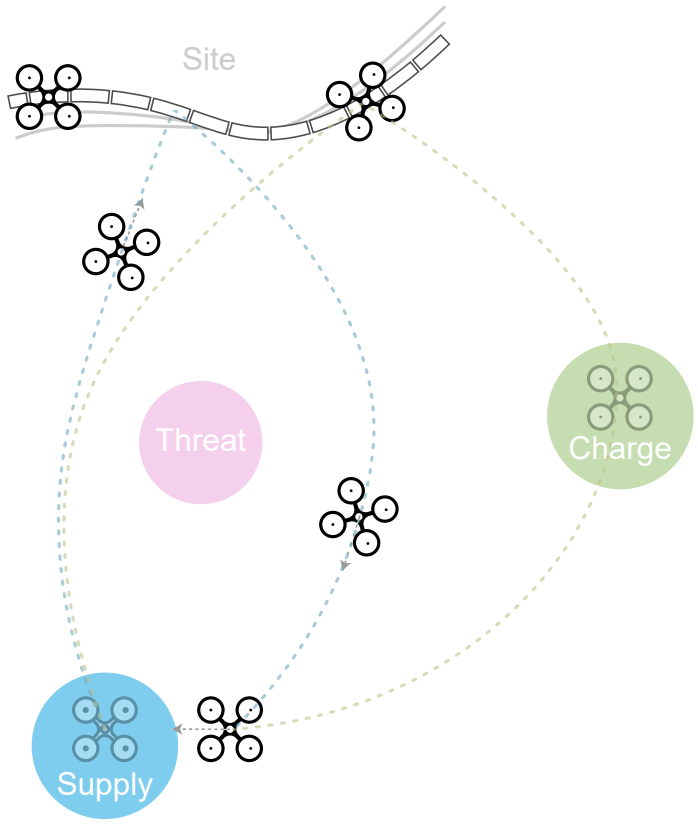


FIGURE 3.1: Multi-drone autonomous construction diagram

3.1.2 Drone tasks

Additive building

Given loaded element/material, the drone needs to deploy it at the target position and orientation. This thesis assumes starting from a position vertically above the target position, the drone is able to achieve additive building by pre-defined macro-actions. A macro-action as used in planning is a meta-action built from a sequence of action steps. Macro-actions should bring the drone to the exact building position and orientation, as well as control drone and end effector during the building process with consideration on

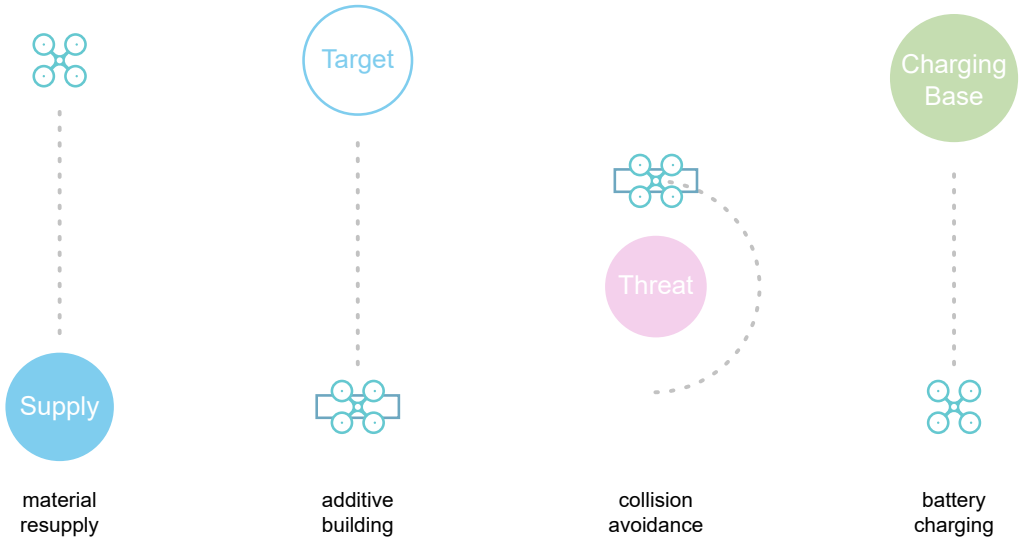


FIGURE 3.2: Drone tasks

building tolerance and finally optionally return to the previous position before building happens. Macro-actions can be either rule-based or RL based or a mixture, depending on the use case. In this thesis, they are implemented as simple rule-based algorithms.

Material resupply

The limited load capacity of drones leads to the necessity of resupplying materials. In a bricklaying scenario, the drone resupplies brick every time after laying, while in a 3D printing scenario, the drone refills 3D printing material after it drains. Similar to the additive building, the process of material refilling is controlled by a rule-based algorithm.

Navigation with collision avoidance

Considering the fact there might be other drones, obstacles, or unflyable areas, drones have to pay attention to surroundings and avoid collisions. For the generic aspect of the system, drones only have access to the lidar sensory input about the environment at execution time. In order to achieve decentralized multi-drone navigation in an unknown environment, deep RL based collision avoidance algorithm is supposed to be implemented in drone's onboard computer.

Battery charging

A drone needs to go to the charging base to charge batteries considering limited battery capacity. Otherwise, they might fail due to battery depletion and need to manually be recovered. The drone should not carry any load which might disturb the charging process. Charging is modeled as a linear process according to charge time, which means the battery level grows at a constant speed.

3.1.3 Server tasks

BIM management

The server should have access to the BIM model, which is the blueprint of the construction. Pre-process job might be needed to parse the model into system readable objects. Moreover, the system should be responsible for monitoring the building progress.

Task assignment

A task indicates a sequence of action steps a drone needs to perform. Other than navigation, rule-based algorithms which can control drone to build, supply, charge, etc. Even though the system doesn't necessarily know the implementation of tasks, since the system has access to the construction progress and all drones' status, it is critical for the system to act as a coordinator to assign proper tasks at the proper timing. Except for task-level assignments, the system should also assign proper targets to drones, such as specifying building elements or supply stations. Optimally assigning tasks is a complex problem, this thesis aims to develop a minimum viable algorithm for simplicity and extensibility.

Drone management

The server keeps tracks of the status of all drones, including position, orientation, task, load, battery, etc. In order for drones to execute macro-actions or navigation, the server should send necessary status information to the drones. Stable communication links are assumed between the server and drones. What's more, the server is designed to be capable of adding or removing drones at runtime, which corresponds to the scalability ability.

3.1.4 Environment

The environment is modeled into a continuous 2.5D world (2D position and orientation with an extra scalar indicating the altitude of entities) with all navigation happens in one 2D plane for the following reasons: First, it simplifies the problem by reducing dimensionality, adding a third dimension may result in 3D behaviors unexpected in the physical world. Second, collision avoidance in the planar world is sufficient for a 3D world. Third, drones navigating in different altitudes may affect each other with air force from propellers which may result in unstable control. A continuous assumption of the environment is much easier to apply to various scenarios since it doesn't require additional discretization of the environment.

While there are several choices of environment simulator frameworks including *OpenAi Gym*[1], *MuJoCo*[2], *Gazebo*[3], this thesis uses a framework built on OpenAi Gym: *Multi-Agent Particle Environment*[4] developed by Lowe et al., 2017; Mordatch and Abbeel, 2017. This framework features a simple multi-agent particle world with a continuous

[1] https://gym.openai.com/
[2] http://www.mujoco.org/
[3] http://gazebosim.org/
[4] https://github.com/openai/multiagent-particle-envs

observation and discrete/continuous action space, along with some basic simulated physics.

Basic quantities of a drone includes: position P, velocity v, acceleration a, external force F, mass m, environment quantities includes: damping ratio ζ, simulation time step dt. The velocity v_i^t and position P_i^t of a drone i at a given time step t is updated as following:

$$v_i^t = v_i^{t-1} * (1 - \zeta) + (F_i^t / m_i) * dt$$
$$P_i^t = P_i^{t-1} + v_i^t * dt$$

3.1.5 Entities

Entities including drones, construction elements, supply stations, threats, and charging stations, are all represented as objects in a programming language. Details of entities are abstracted since they differ from case to case, only the behavior is modeled at a high level. Each entity is modeled as a circle in the simulator since it is easy to represent a circle using only two notations(center and radius) and calculate distances between entities. Entities are expected to have the following common attributes: size, state(2D position, 2D velocity, altitude, orientation), mobility, and collidability (whether the entity can collide with others).

3.1.6 Drone

Dynamics and control schemes

A drone(quadcopter) can be modeled as a rigid body composed of a frame and four single rotor thrusts. In the real world, the low-level control of a quadcopter(drone) is to directly adjust the power of four rotors. In this way, the drone can perform roll, pitch, yaw, and altitude changes (Fig. 3.3). However, projected to a 2.5D world, 3D movements are simplified to 2D position and rotation changes along with an independent attitude change. This way of simplification also makes it possible to be extended to various types of drones such as quadcopters, hexacopters, and octocopters.

This thesis assumes continuous control of the drones, which means the drones state, including position, rotation, velocity, and acceleration can be any reasonable real number. Although it is common to discretize action space, a continuous control method can control drones in a more decent way and can easily be adapted to discrete control, however, discrete control is not so easy to be generalized to a continuous version. This implies continuous control is more generalized than discrete control.

Control of a drone can be categorized into two schemes: supervised control and unsupervised control. For the supervised control, the drone is controlled by deterministic rule-based algorithms. In the unsupervised case, where the drone utilizes the onboard RL model to navigate by a stochastic policy, which receives observations from the environment and the server then outputs a 2D force, which can be converted to translational acceleration, to control the velocity of the drone. During navigation, the orientation of the drone is simplified as velocity direction and altitude remains unchanged. Both rule-based algorithms and RL model are implemented in the onboard computer of the drone, while the server allocates resources and assign tasks.

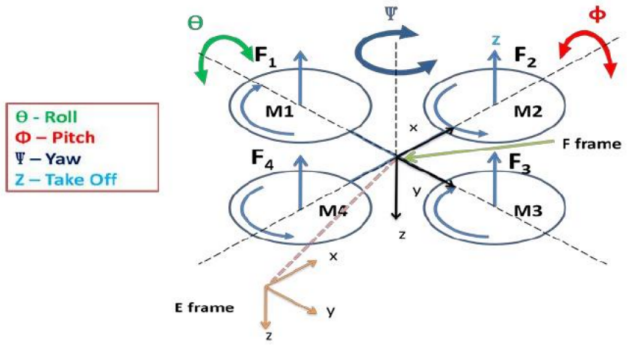


FIGURE 3.3: Quadcopter Control Frames (Kurak and Hodzic, 2018)

Localization

Since drones can be viewed as a rigid body, it is possible to use motion capture to estimate its coordinate in high accuracy for indoor spaces in most researches. During navigation, the server keeps track of all drones' position and orientation by motion capture calculates the difference between goal position, and sends them to corresponding drones.

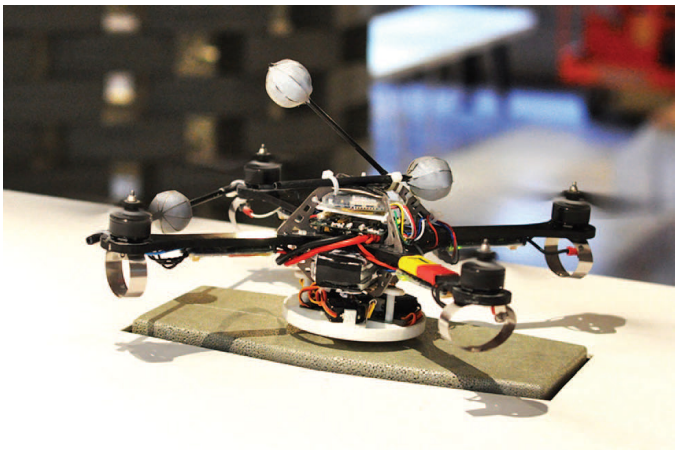


FIGURE 3.4: A drone with motion capture markers (Augugliaro et al., 2014)

Battery

The drone's operation time duration depends on its battery. For small size drones, their battery capacity is not large enough for long-time operation. As a result, it is necessary to model the battery consumption and charging process. This thesis assumes linear battery consumption and charging speed, according to time. The battery is represented as a numeric attribute of the drone object in a programming environment.

End effector

Controlling end effectors such as gripper or extruder is modeled as a defined rule-based action while the end effector itself is not modeled or represented explicitly in the system. Abstracting end effector simplifies the problem and provides extensibility for drones to perform different types of construction-related jobs.

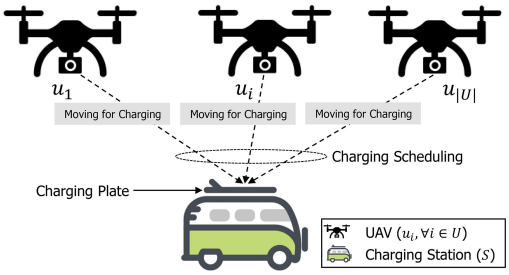
3.1.7 Construction elements

As reviewed in 2.2.2, there is a wide choice of construction elements. This thesis represents construction elements as entities for management convenience. Besides, carrying an element does is assumed not to apply any physical force to the drone. To be noticed, elements might have a dependency on each other, which should be considered when the server assigns tasks to drones. For instance: the drone can't build a totally hovering element if its supporting elements are missing. Moreover, the elements might not necessarily be the same. Different elements can exist, the system should assign the correct target position for different elements.

3.1.8 Charge station

Charge stations could be set up not only considering the short battery life of drones but provide a docking place for drones that are not active (when there is no task allocated for the drone). Although wired charging is available for drones, it requires a connection of wires, which is hard to automate. Thus, wireless charging is more popular in most cases. Charge station can be one-to-one charging (Fig. 3.5a) or one-to-many charging (Fig. 3.5b). This thesis uses the one-to-one charge stations so that the number of drones matches the number of charge stations.

(A) one to one charging
(Augugliaro et al., 2014)



(B) one to many charging
(Shin, Kim, and Levorato, 2019)

FIGURE 3.5: different charging methods for drones

3.1.9 Supply station

Supply stations possess enough elements for drones to build. In the scenario of bricklaying, drones pick up bricks. While in the scenario of 3D printing, drones refill 3D printing material. The design of supply station involves consideration of elements and end effectors: how the end effectors refill the elements. The refill process could be autonomous or manually performed by workers. The system is designed to be able to manage different

multiple supply stations for the following two reasons. First, there might exist different buildable elements, thus letting one supply station supply one kind of element that is more manageable. Second, the shortage of supply stations may lead to multiple drones competing and waiting, which lowers the construction efficiency.

3.1.10 Waiting area

The waiting area is a buffer area to address the problem of resource competing among drones. A drone enters a waiting area when another drone occupied the space it wants to access. The waiting area can be designed around the construction area and supply station. In order to make drones lining at the waiting area, not only free areas need to be designed, but an algorithm is needed to manage the order of waiting drones. 4.2.3 showed a possible design of the waiting area for the supply station.

3.1.11 Threats

Threats are areas where drones need to avoid, as shown in Fig. 2.12a. In the context of construction, threats can represent obstacles, architecture columns, walls, or even workers. These threats are usually sparsely distributed if the construction site is relatively clear. In order to model an uncertain dynamic environment, the threats can change size and position at execution. This thesis assumes threats to be infinitely high cylinders.

3.1.12 Unmodeled part

As described in this section, each part of the problem has levels of abstraction to make the proposed system be generic. There are some aspects that are not modeled but need to be implemented in real-world applications either because the problem is hard to represent or solve in the simulator or because the problem should be solved case by case. First, hardware and objects are assumed to function well in this thesis, however, they need to be carefully designed and implemented. Second, drone control is simplified as 2.5D control by force, acceleration, orientation, and velocity. A more sophisticated control scheme should be implemented to control a real drone. Third, environment uncertainty or unexpected behaviors or failures are not sufficient to cover all real-world situations although the system tries to cope with some of the uncertainties. Fourth, the construction process is simplified as simple assembly or material deployment, which is not enough for a real-scale construction. Fifth, the cost of the system is not analyzed. Finally, human interference, like operators and workers should be considered to let users properly interact with the system.

3.2 Multi-drone autonomous construction system

This thesis focuses on developing such a software framework for multi-drone autonomous construction that is scalable and extensible enough to be used in various scenarios. Scalable refers to in one kind of application scenario, the system can handle different numbers of drones, building elements involved in the blueprint, other entities, from a few to dozens without making great changes. Extensible means the system allows developing extensions for extra functionalities to be applied in various scenarios. The extension

could be the implementation of new functionalities or improvements on existing algorithms.

3.2.1 Features

Autonomous

Autonomous drone construction refers to minimum human involvement in the construction process. The system achieves autonomy in two ways: an RL-based algorithm for navigation and rule-based algorithms for task assignment and execution. Admittedly, there are some procedures that can't be automated, such as system development, environment setup, drone setup, failure recovery, etc.

Scalable

The system should support various numbers of drones, from a few to dozens, as well as the various scale of the construction without large modification. In this thesis, scalability is achieved by the implementation of algorithms. For smooth deployment, change in scale should be accompanied by environment setting changes (e.g. more supply stations, charge stations, space define), due to resource allocation issues.

Extensible

The system should be able to be generalized to different kinds of construction scenarios, which is the reason of problem abstraction and framework-like implementation. By proper modeling, the construction problem is decomposed and solved part by part, which is also good for future extension development. Although the system uses rule-based algorithms to implement some typical macro-actions or tasks, they are done in a minimum viable manner and should be extended to become more robust in real-world usage, which differs case by case.

3.2.2 System architecture

The multi-drone autonomous construction system consists of a central server, distributed onboard algorithms, including deep RL navigation model and macro-actions for other tasks and an interface for design software integration. However, since this thesis is built on a simulator environment, all onboard algorithms are implemented and computed in the server for this thesis. The system architecture is illustrated in Fig. 3.6.

For the server, there is a driver script controlling the main loop of the simulation, where it obtains actions by inputting agents' observation into the navigation model, then steps the simulation environment to acquire new observation. The Rhinoceros integration part is mentioned in 3.4.

There are two scenarios existing in the environment. One is for training the other is for executing the construction tasks. The entities in the training scenario are simplified into 2D particles since this is enough for the RL training setup.

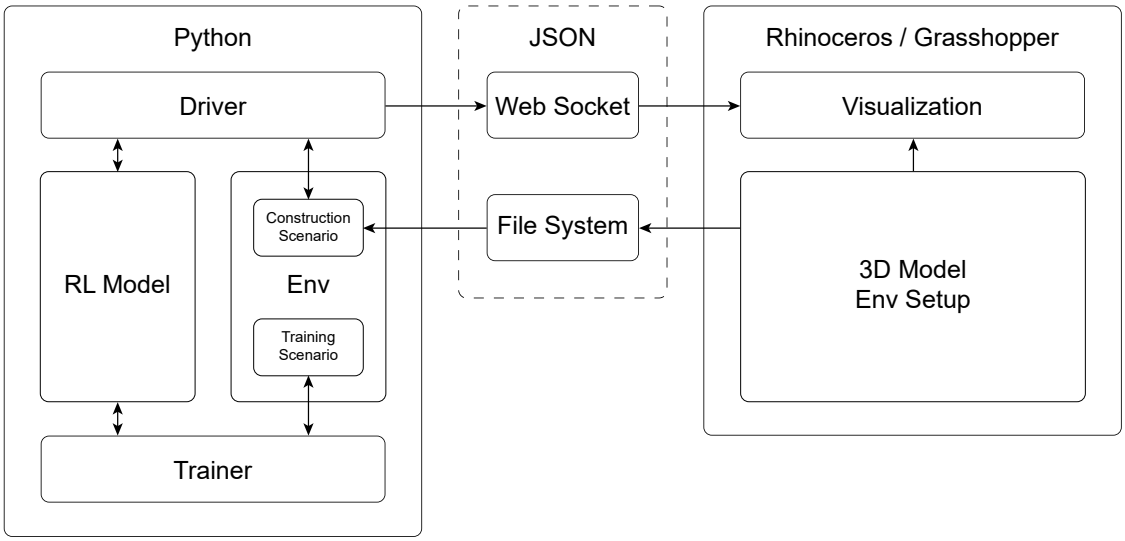


FIGURE 3.6: System architecture

3.2.3 Entities

Entities mentioned in 3.1.5 are all represented by classes inherited from the *Entity* class in *Multi-Agent Particle Environment*. Some key attributes/methods of these classes are designed as below:

Drone

- *size*: agent radius
- *state*: current position, velocity and orientation
- *previous state*: state of the previous time stamp, used to calculate reward
- *hidden state*: hidden states for LSTM layer input
- *lidar*: a few lidar ranges in recent time stamps
- *target*: the target state agent needs to navigate to
- *load*: current construction elements the agent is carrying
- *capacity*: the largest number of elements the agent can carry
- *dock*: the charging station the agent is assigned to
- *battery*: the remaining battery
- *functions*: a list to store functions waited to be executed in a supervised manner
- *resupply()*: a rule-based function to control the agent to refill material
- *build()*: a rule-based function to control the agent to deploy material

- *charge()*: a rule-based function to control the agent to charge
- *wait()*: a rule-based function to control the agent to wait

Construction element

- *state*: current position and orientation
- *target state*: target position and orientation
- *type*: type of the element
- *parents*: a list of other elements the element depends on

Supply station

- *state*: position and orientation
- *agent*: current agent which is refilling at the supply station
- *type*: the type of element supply station possesses
- *buffer*: *Supply buffer* of the station

Supply buffer

- *positions*: a list of position and orientation for agents to wait at
- *agents*: list of agents in the queue
- *enqueue()*: add an agent to the queue and assign a position for it
- *dequeue()*: pop the earliest entered agent

Charge station

- *state*: position and orientation

Threat

- *state*: position and orientation
- *size*: threat's radius

3.2.4 Simulation algorithm

Algorithm 1 is a high-level pipeline of the central server. After initialization, the system goes into a loop of stepping agents and assigning tasks accordingly. Due to the simulator environment, it is actually the server interacting with the simulation environment where agents step rather than agents step by themselves. At the end of the loop, server checks if the config has changed or not to add/reduce drones or update entity attributes.

Algorithm 1 Multi-drone autonomous construction system pipeline

```
initialize environment, entities
initialize BIM model from config files
while not terminated do
    for each drone do
        collect observation
        if drone is unsupervised then
            step the RL model
            store hidden states
        else
            step the supervise function(s)
            if task finished then
                set drone to unsupervised
                update drone's target accordingly
            end if
            continue
        end if
        if drone reaches target then
            set drone to supervised
            assign corresponding supervise function(s) to drone
        end if
    end for
    for each supply station do
        if station is occupied and buffer is not empty then
            buffer dequeue
        end if
    end for
    update BIM model based on built elements
    if config files updated then
        update system config
    end if
end while
```

Macro-actions, that control the drone to perform tasks other than navigation, are implemented in minimum viable rule-based algorithms in this thesis. In real-world applications, they should be implemented in drones' onboard computers to interact with the physical world. They could be either rule-based or stochastic but should be robust enough to cope with precision problems and errors.

3.3 Deep RL navigation algorithm

3.3.1 Problem formulation

The multi-drone navigation problem in the context of autonomous construction is defined as a multi-agent holonomic collision avoidance on a 2D plane. Drones are modeled as homogeneous agents that share the same policy. The agent has no access to

the exact system state, instead, it maintains a probability distribution over the set of possible states. Moreover, although there are multiple agents involved, all agents are exchangeable and act independently, it is essentially a single agent decision-making problem. Such problems can be formulated as a partially observable Markov decision process (POMDP) (Cassandra, 1998). Formally, a POMDP can be represented by a 6-tuple: $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O})$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T}$ is a set of conditional transition probabilities between states, $\mathcal{R} : \mathcal{S} * \mathcal{A} \to \mathbb{R}$ is the reward function, Ω is a set of observations and $\mathcal{O}$ is a set of conditional observation probabilities.

At each time step t, the i-th drone observes o_i^t and computes an action a_i^t to navigate to the goal. o_i^t is drawn from $\mathcal{T}(s_i^t)$, which only provides partial information about s_i^t, since the drone is modeled to be unaware of other drone's states and intents. The observation is mainly based on sensory input and does not require perfect sensing. This partial observation setting is more applicable and robust in real-world applications. (Long et al., 2017)

The objective of policy optimization is to minimize the expected mean arrival time of all drones in an episode, defined as:

$$\underset{\pi_\theta}{\text{argmin}} \ \mathbb{E}[\frac{1}{N}\sum_{t=1}^{N} t_i^g \mid \pi_\theta]$$

3.3.2 RL setup

Observation space

Similar to Lowe et al., 2017, the observation space of the i-th drone at a given time step t, o_i^t consists of 3 components: $[o_z^t, o_g^t, o_v^t]$. To be specific, $o_z^t \in \mathbb{R}^{3*256}$ is the 3 recent time frames of 256 distance arrays (Fig. 3.7), similar to a lidar. $o_g^t \in \mathbb{R}^2$ is the x, y relative position between drone and its target goal. $o_v^t \in \mathbb{R}^2$ is the x, y translational velocity of the drone. In reality, o_g^t is given by the server or computed by drone itself knowing position of the goal and itself from motion capture system, o_v^t is estimated by onboard sensor and processor.

Action space

The action space a^t of a drone is the force applied on it in x, y axes, i.e. $a^t = [a_x^t, a_y^t]$. Since drone mass is modeled as a certain number in the environment, the action space is equal to the translational acceleration multiplied by a mass scalar. Considering a stable control, a^t is clipped to be between $[-1, 1]$.

Reward design

In order to minimize the mean arrival time of all drones, the reward function is designed as a sum of goal reward and collision reward:

$$r_i^t = ({}^g r)_i^t + ({}^c r)_i^t$$

where:

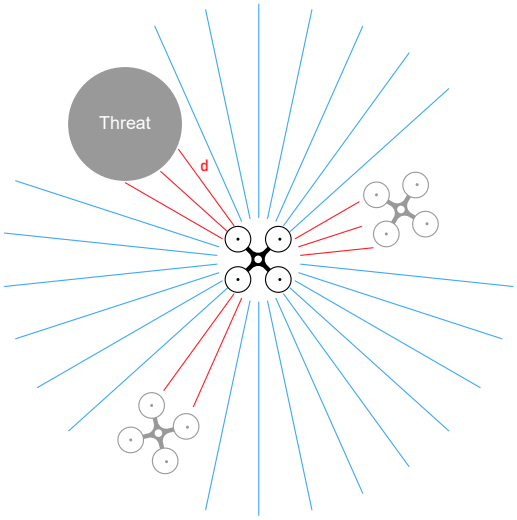


FIGURE 3.7: Lidar observation

$$({}^{g}r)_t^t = \begin{cases} r_{arrival} & \text{if } \left\| p_i^t - g_i \right\| < d_{tolerance} \text{ and } \mid v_i^t \mid < v_{threshold} \\ \omega_g \left\| p_i^{t-1} - g_i \right\| - \left\| p_i^t - g_i \right\| & \text{otherwise} \end{cases}$$

$$({}^{g}r)_t^t = \begin{cases} r_{collision} & \text{real collision} \\ (1 - d/d_{pseudorange}) * r_{collision} & \text{pseudo-collision} \\ 0 & \text{otherwise} \end{cases}$$

The drone is awarded if it reaches its target or makes an approaching step towards it. While it is penalized for getting away from the target or collision. For the criteria of reaching the target, the distance between the drone and its target should be within a tolerance distance and the speed of the drone should be lower than a threshold. This ensures drones not overpassing the target because of inertia.

A pseudo-collision mechanism (Fig. 3.8) is adopted to prevent drones from getting to close from each other. Different from Qie et al., 2019, only the pseudo-collision range of the agent itself is considered.

Policy network architecture

Given a sensor-based input, the policy is expected to compute a control force for collision avoidance navigation. A deep neural network is used to approximate the policy. Two 1D convolution layers and a dense layer are employed to extract features from lidar observation o_z^t, the result is then concatenated with o_g^t and o_v^t and passed through a dense layer, an LSTM layer and another dense layer to output predicted translational acceleration a_{mean}^t. The activation function of the last dense layer is *tanh* while the other layers are all *ReLU*. In order to achieve exploration, the output of the action is drawn from a Gaussian distribution $a^t \sim \mathcal{N}(a_{mean}^t, a_{logstd}^t)$, where a_{logstd}^t is trained separately.

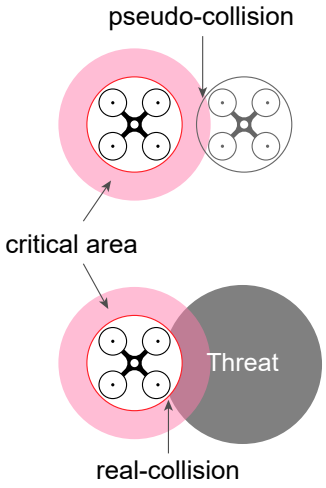


FIGURE 3.8: Pseudo-collision and real collision

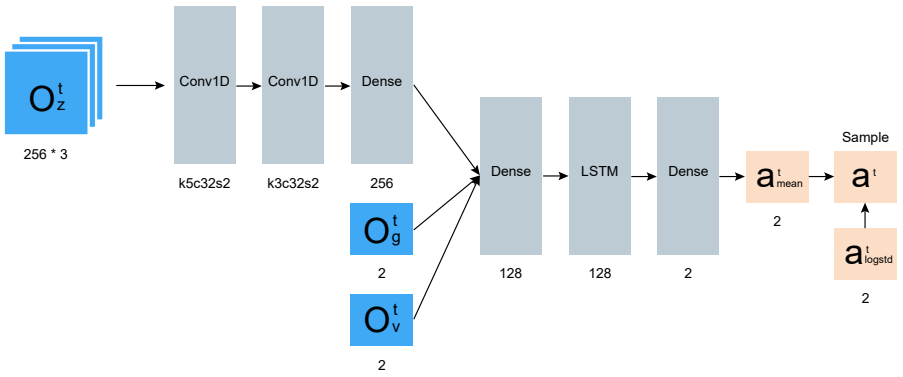


FIGURE 3.9: Policy network architecture

3.3.3 Training algorithm

This thesis uses a popular policy gradient algorithm, Proximal Policy Optimization (PPO) (Schulman et al., 2017) to train the RL model. PPO bounds the parameter update to a trusted region to improve stability. PPO has been tested on various benchmark tasks and proved to produce satisfying results with greater simplicity. The approach follows the *centralized learning, decentralized execution* paradigm. The policy is shared among all drones and trained with experiences collected online by all drones.

Algorithm 2 describes the training logic. Adapted from Schulman et al., 2017; Long et al., 2017, this algorithm uses the actor-critic PPO with the clipped surrogate objective on multiple robots. The actor-critic framework implies an actor (policy) network is used to compute the best actions, the critic network, on the other hand, is used to approximate the value function. First, the environment and models are initialized for experience collection. After iteratively feeding new observations to the agents simultaneously, a set of trajectories, containing $\{o_i^t, r_i^t, a_i^t\}$ are collected and used to calculate generalized advantage estimation (GAE). Then policy network π_θ and value network V_ϕ parameters

are updated by PPO loss and MSE loss using ADAM optimizer and their parameters are not shared. Using two separate networks is reported to have better results in Long et al., 2017. The experience collection and network updating form an epoch, by repeating training for enough epochs, the policy can be learned.

Algorithm 2 Actor-Critic Clipped Objective PPO with Multiple Drones

Initialize policy network π_θ and value network $V_\phi(s_t)$
for iteration $= 1, 2, \ldots,$ **do**
  for t=1,2,...,T **do**
    for drone $i = 1, 2, \ldots, N$ **do**
      Run policy on the drone π_θ
      Store $\{o_i^t, r_i^t, a_i^t\}$ in memory
    end for
    Estimate advantages using GAE $\hat{A}_i^t = \sum_{l=0}^{T} (\gamma\lambda)^l \delta_i^t$,
    where $\delta_i^t = r_i^t + \gamma V_\phi(s_i^{t+1}) - V_\phi(s_i^t)$
  end for
  $\theta_{old} \leftarrow \theta$
  Sample a batch with size S from the memory
  for $j = 1, 2, \ldots, S$ **do**
    $L^{PPO}(\theta) = \hat{\mathbb{E}}_t[\min(\frac{\pi_\theta(a_i^t \mid o_i^t)}{\pi_{old}(a_i^t \mid o_i^t)}\hat{A}_i^t, \text{clip}(r_i^t, 1-\epsilon, 1+\epsilon)\hat{A}_i^t)]$
    $L^V(\phi) = -\sum_{t=1}^{N}\sum_{t=1}^{T_i}(\sum_{t'>t}\gamma^{t'-t} r_i^{t'} - V_\phi(s_i^t))^2$
    Update θ with lr_θ by Adam w.r.t $L^{PPO}(\theta)$
    Update ϕ with lr_ϕ by Adam w.r.t $L^{PPO}(\theta)$
  end for
end for

Collaborative transportation

Although this RL algorithm is based on a single agent. Although not implemented, there is still possibility to incorporate multi-agent behavior such as transportation by augmenting observations. Fig. 3.10 illustrates a collaborative transportation diagram, by treating collaborating drones as a single agent, the algorithm is still expected to work.

3.4 Design software integration

With integration in design software, it becomes much easier for designers to experiment and explore the new possibilities of aerial construction. For a parametric modeling software, 3D model parameters could be directly passed to the system to set up the environment without any requirement of programming knowledge.

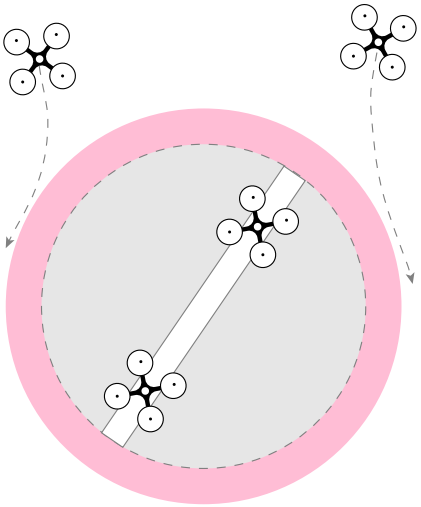


FIGURE 3.10: Collaborative transportation

3.4.1 Rhinoceros

Rhinoceros is a powerful 3D modeling software widely used in architecture, industrial design, product design, etc. Apart from two scripting languages, Rhinoscript and Python, Rhinoceros supports, It also has a plug-in, a parametric modeling programming tool, Grasshopper, with a large architect user base due to the ability for parametric modeling and its easy-to-use GUI.

In order to provide a seamless experience for designers, this thesis developed a script in Grasshopper to integrate the base system developed. In the interface of Rhinoceros/Grasshopper, users can define BIM blueprint for the system to simulate, view realtime 3D progress, obtain states and actions of all entities and make realtime adjustments which can be instantly reflected in the simulation.

3.4.2 Method

First, the 3D BIM model is modeled in Rhinoceros or Grasshopper. Then the model is translated into system readable data in Grasshopper. The attributes need to articulate typically include target position, target rotation, type, and dependency. Other entities such as charge stations, supply stations, and threat areas. The config data is written into a JSON file and can be updated in the process of execution. After the server initialized the environment config, it runs the simulation and sends JSON data back to the Grasshopper for visualization through a web socket. The data includes information about drone states and brick states.

3.5 System pipeline

Fig. 3.11 shows how a user interacts with the system and gives a general idea of the system pipeline. Basically, the user needs to run the driver script at the background and defines model, environment settings and entities in Grasshopper, the config is passed

to the driver and the driver automatically calculates drone behavior and position, along with construction progress step by step. After each step, the driver passes the result to Grasshopper for visualization. During the process, the user can adjust settings as he/she wants and the result will reflect on the next step.

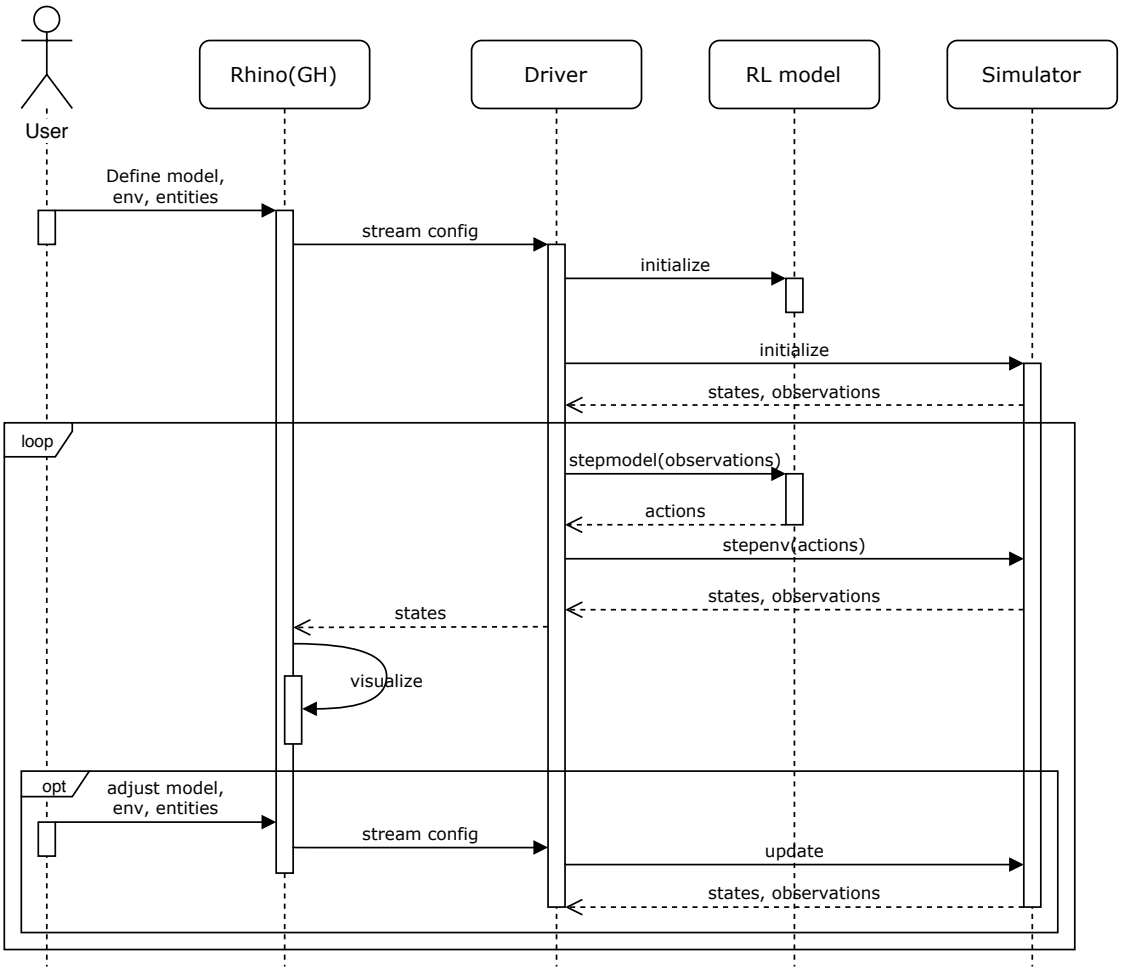


FIGURE 3.11: System sequence diagram

3.6 Applicable scenarios

Based on case study in section 2.2.2 and drone based construction, a few applicable scenarios are suggested. It is believed with further development in drone technology, drones can accelerate these construction tasks. Moreover, as the system is designed to be generic, some other construction tasks can also be automated by drones under certain setup.

3.6.1 Discrete building material assembly

Discrete building material assembly is the most common tasks in construction and the best applicable scenario for the system this thesis is proposing. Prior works (Lindsey,

Mellinger, and Kumar, 2012; Augugliaro et al., 2014; Wood et al., 2018) have already shown promising future of this scenario. With a team of drones, tasks like bricklaying, frame structure assembly can be automated in a much easier and faster way as long as hardware suffices. Despite the fact that the system is mainly built in an asynchronous manner, it is still possible to incorporate synchronous behavior to become a mixed setting. With an additional algorithm controlling synchronous drones for cooperative construction, other drones that are controlled by the autonomous navigation algorithm still work without modification because the algorithm is based on sensory input.

3.6.2 3D printing

Equipped with an extruder, a drone can act as a mid-air 3D printer. Compared to elements like bricks, 3D printing is lightweight and material cohesion allows more complex structure. Hunt et al., 2014 points out scenarios of drone-based 3D printing, "Potential applications include ad-hoc construction of first response structures in search and rescue scenarios, printing structures to bridge gaps in discontinuous terrain, and repairing damaged surfaces in areas that are inaccessible by ground-based robots." The potential difficulty drone-based 3D printing faces apart from precision is path design. The printing path should be carefully designed to allow pause and resume of printing procedure because of the limited load capacity of material as well as to allow multiple drones print at the same time without path collision.

3.6.3 Facade coating

Drones can reach places where common robots can't, so it gives drone-based autonomous facade coating over other robots. Equipped with a sprayer, drones can spray materials onto the facade. Compared with 3D printing, facade coating does not require such high precision. Material supply can be designed as a mobile storage tank carried by the drone instead of a pipe connected with external storage so that multiple drones can easily work together without concerning pipe twisting.

Chapter 4

Experiments

In this chapter, the methodology developed is examined in applications. The multi-drone autonomous construction system serves as a software framework, in order to use in applications, additional implementations should be developed as an extension of the framework. This chapter first describes the training process of the RL model for multi-drone navigation. Then two experiments, bricklaying and facade coating, are implemented for simulation based on the framework and 3D modeling software integration.

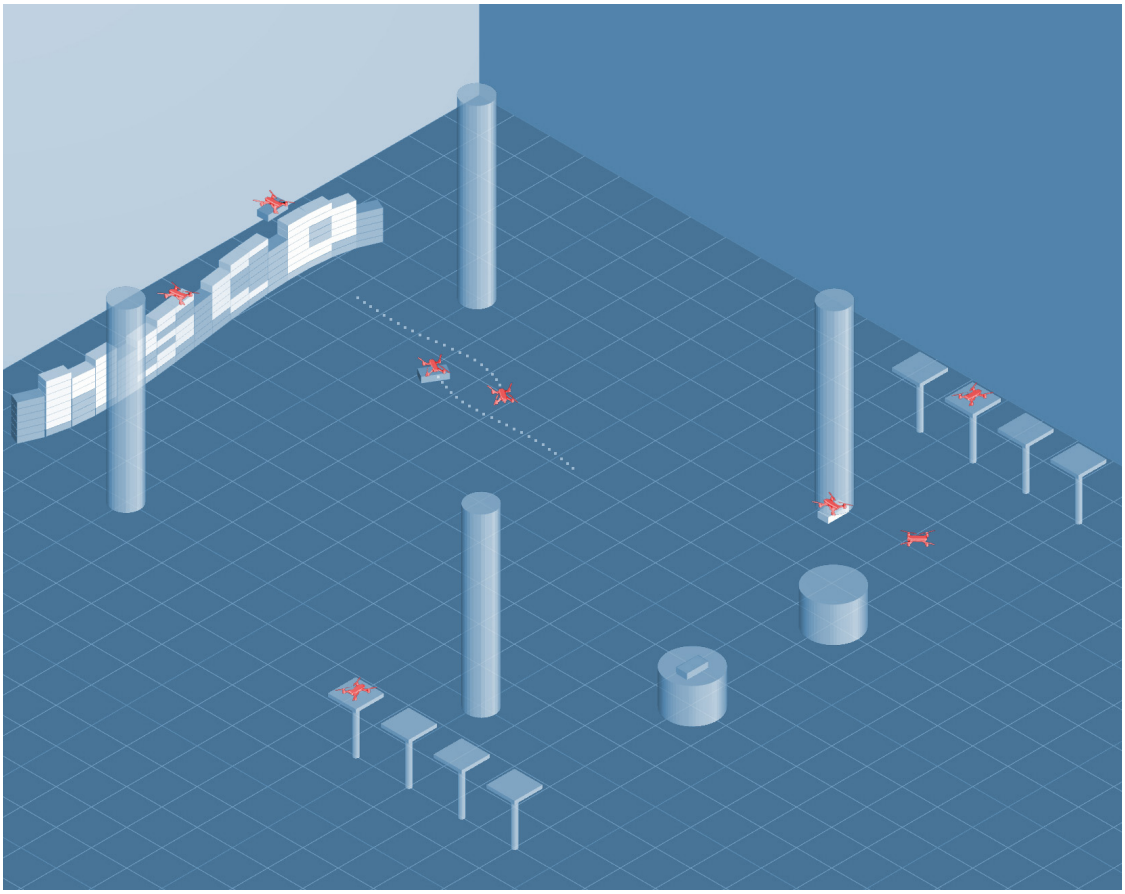


FIGURE 4.1: Brick laying illustration

4.1 RL model training

This section introduces the training setting and process of the RL algorithm. First introduced the two-stage training methods to train in a curriculum learning fashion. Configurations of the simulation environment and hyper-parameters are showed. Finally, the result is reviewed.

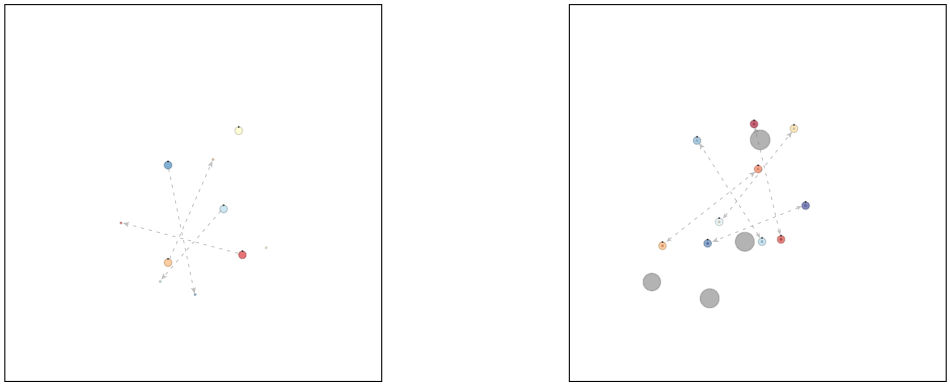
Training stages

Similar to Long et al., 2017, this work uses a two-stage training method to learn the policy in a curriculum learning fashion, in which policy network is reported to converge faster. The first stage (Fig. 4.2a) is 5 agents in a randomly generated scenario based on generation algorithm 3 in order to have a similar target distance for each agent.

Algorithm 3 Stage1 initialize algorithm

for each drone **do**
  generate a agent pos $P_x, P_y \sim \mathrm{U}(-d, +d)$
  while distance$(P, origin) > d$ **do**
    generate a pos $P_x, P_y \sim \mathrm{U}(-d, +d)$
  end while
  generate a target pos $T_x, T_y \sim \mathrm{U}(-d, +d)$
  while distance$(T, origin) > d$ or distance$(T, P) \notin (d-1, d+1)$ **do**
    generate a target pos $T_x, T_y \sim \mathrm{U}(-d, +d)$
  end while
end for

In the second stage (Fig. 4.2b), 10 agents are trained in a scenario with a number of sparsely distributed threats. After generating threats, 5 agents are generated using algorithm 3, another 5 agents are generated at the goal position of the previous 5, and their target positions are the goal positions of the previous 5 agents. Generating stage in this swapping way makes agent avoid more robustly from a potential head-to-head collision.



(A) First stage initialization (B) Second stage initialization

FIGURE 4.2: different charging methods for drones

Training configuration

Table 4.1 is a list of hyper-parameters used in training Algorithm 2. Besides, the environment is set as a canvas of size 10 by 10. The simulation time step dt is set as 0.08. Each agent has a size between 0.25 to 0.35 and has a lidar range of 4.

Parameter	Value
λ	0.95
γ	0.99
T	80
S	1024
ϵ	0.1
lr_θ	$5e-5$ (first stage), $2e-5$ (second stage)
lr_ϕ	$1e-3$ (first stage), $4e-4$ (second stage)

TABLE 4.1: Hyper-parameters

Results

The training was done on a computer with i7-6700K CPU and an Nvidia GTX 1080 GPU. Fig. 4.3 showed the episode reward averaged across agents. The first stage was trained for 60 hours and the second trained for 230 hours. Although the policy was already robust at 80 hours of training in the first stage, it was continued trained for marginal improvements. The significant time duration of the second stage is because of the computation burden for calculating the lidar of more agents. Besides, the initial reward drop observed in the second stage is because the second stage has more agents and threats while the episode length is the same as the first stage. However, in the second stage, the agents learn to slow down when reaching the target, which leads to an increase in reward.

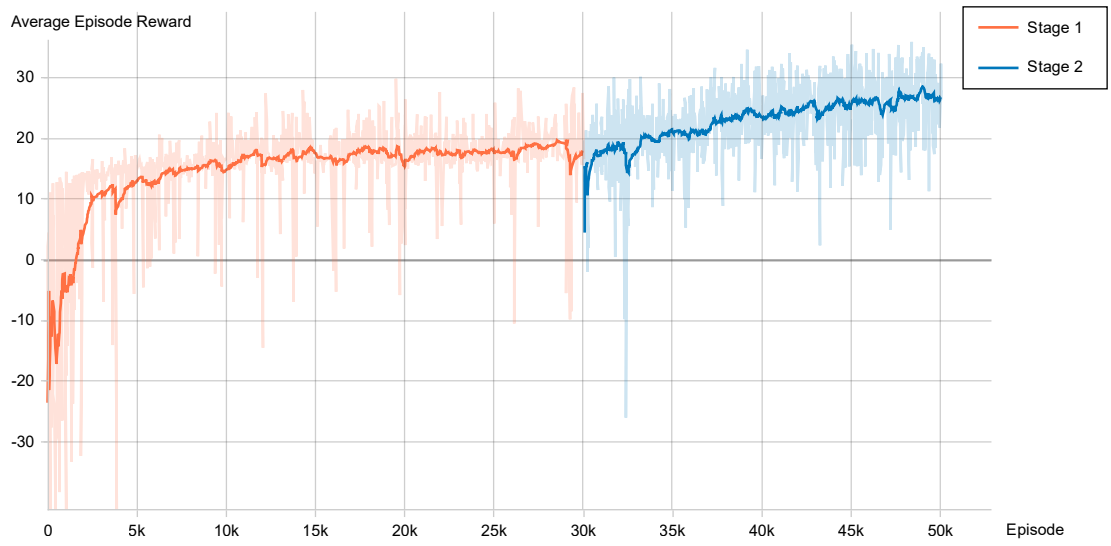
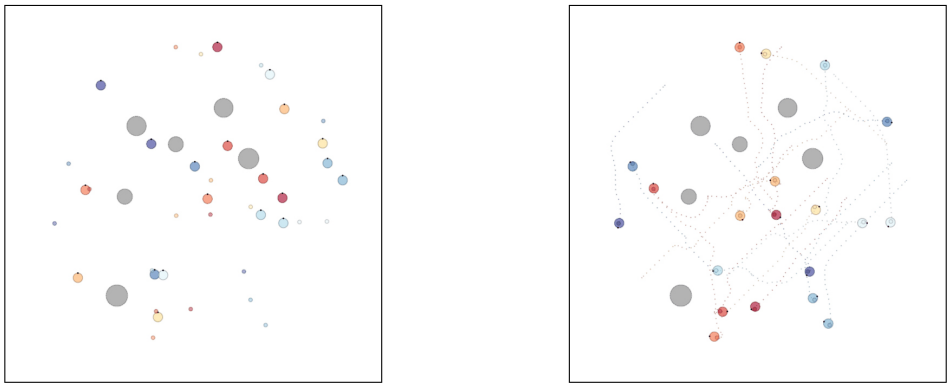


FIGURE 4.3: Average episode reward (smoothing=0.95)

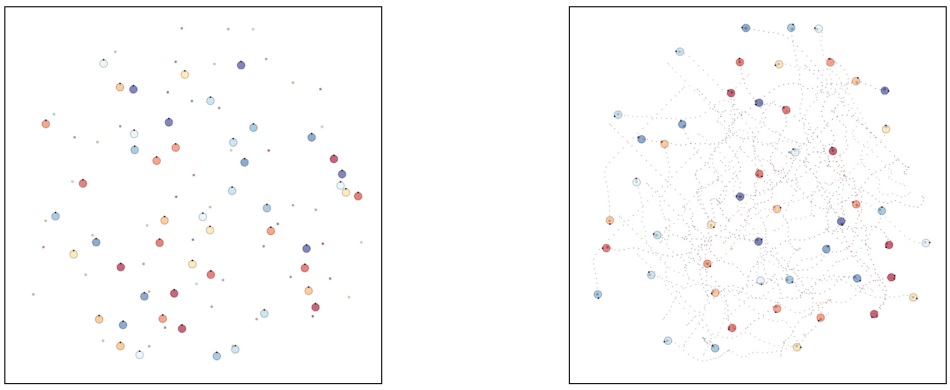
The learned policy is then tested on a randomly generated scene with 20 agents (Fig. 4.4). All 20 agents successfully reached their targets without any collision, even though the policy is trained on 10 agents. Next, 50 agents are tested in the same initialization range without threats and managed to reach targets collision-free (Fig. 4.5). Moreover, the system can incorporate any large number of agents if the environment is large enough. The result shows good scalability of the algorithm and generalizability to different scenarios. Besides, it is not sensitive to environmental turbulence and dynamic obstacles. As the training settings imply, the policy also supports randomly initialized agents, targets, and obstacles. However, due to the reward design, the agents tend to be conservative to high-risk moves, i.e. space is too narrow, because collision and near-miss and are largely penalized for safety.



(A) Initialization for 20 agents

(B) Trajectories for 20 agents

FIGURE 4.4: Policy evaluated on 20 agents



(A) Initialization for 50 agents

(B) Trajectories for 50 agents

FIGURE 4.5: Policy evaluated on 50 agents

4.2 Brick laying

In this experiment, a simulation experiment of using the multi-drone autonomous construction system to build a curvature brick wall is designed and conducted. First, the

blueprint model is designed, then special mechanisms are implemented for the bricklaying scenario. Finally, the simulation process and results are shown.

4.2.1 Blueprint

A curvature brick wall (Fig. 4.6) is modeled in the parametric modeling software Rhinoceros with the help of the Grasshopper plugin. The brick wall consists of two types of bricks, light gray bricks, and dark gray bricks, to form an "MSCD" word on the wall. Since the wall is modeled in a parametric way, Grasshopper has information on bricks' position, orientation, type, and dependency which can directly be input to the system.

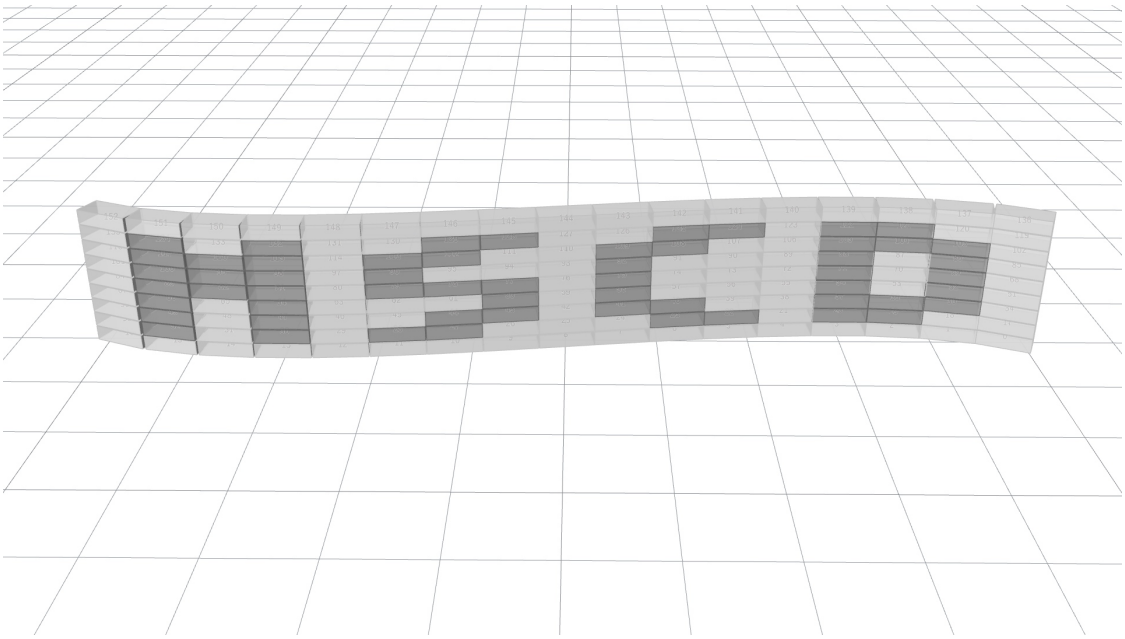


FIGURE 4.6: Brick laying blueprint

4.2.2 Environment setup

As shown in Fig. 4.7, Drones are deployed to build the brick wall, each is initialized at a charge station. Three supply stations, including two light gray brick stations and one dark gray brick station, are set at the other side of the site, each is accompanied by a waiting area. Four threat area is set to represent columns. Through trails, eight is found to be enough for the number of drones for this blueprint and setup. More drones will cause resource competition and result in a waste of time for waiting.

4.2.3 Brick laying implementation

Based on the framework of the multi-drone autonomous construction system, additional logics are designed to implement the bricklaying process.

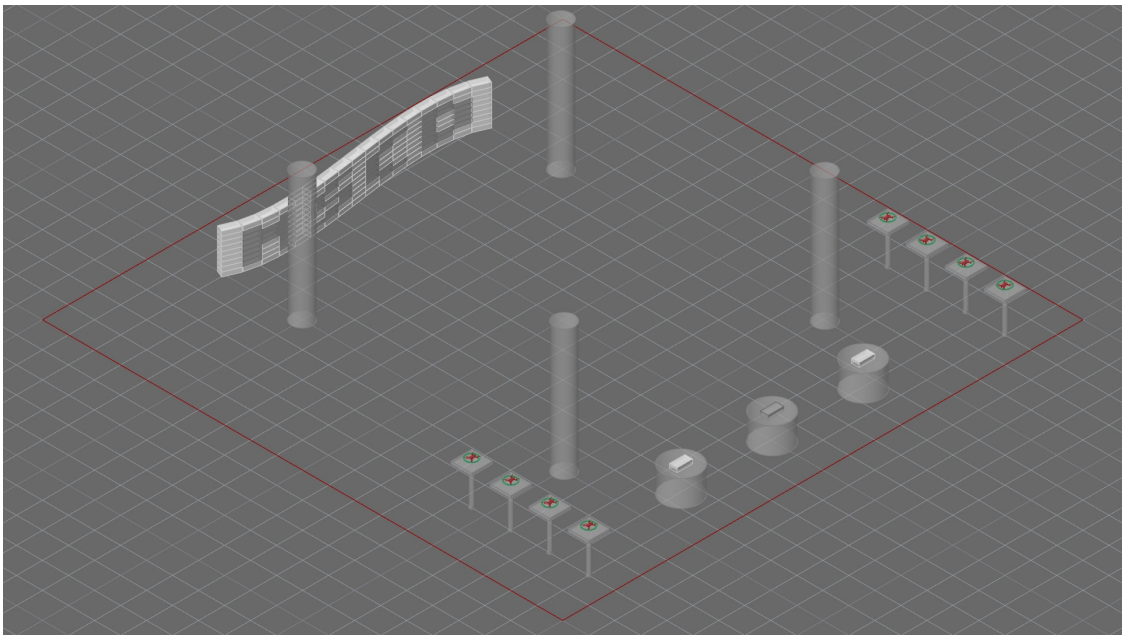


FIGURE 4.7: Environment setup for brick laying

Brick dependency design

The easiest way for building order is to completely following the brick order, one by one and row by row. However, it is not safe for two drones to lay adjacent bricks at the same time. Other row-by-row solutions might also have edge cases. Therefore, in order to have a more uniformly distributed building style while avoiding collisions, a node-based algorithm is designed. For a given brick, the parent bricks can be defined as a number of supporting bricks on a lower row as illustrated in Fig. 4.11. To be mentioned, the even-numbered bricks in a row are parents of their adjacent odd-numbered bricks. Defining such in-row dependency ensures two adjacent bricks will not be built simultaneously to avoid collisions. A detailed algorithm regarding initialization and bricklaying is described at Algorithm 4.

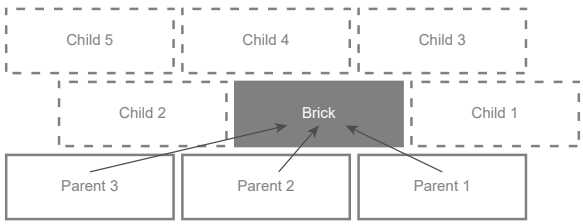


FIGURE 4.8: Brick dependency diagram

Algorithm 4 Brick laying algorithm

```
1:  function INITIALIZE
2:      Initialize two global empty sets: buildable, built
3:      for each brick object do
4:          initialize two empty sets: brick.parents, brick.children
5:          brick.parents ← all dependent bricks
6:          if brick.parents is empty then
7:              add brick to buildable
8:              continue
9:          end if
10:         for each parent in brick.parents do
11:             add brick to parent.children
12:         end for
13:     end for
14: end function
15: function LAYBRICKS
16:     while buildable is not empty do
17:         sample a brick and remove it from buildable
18:         assign a drone to lay the sampled brick
19:         after the brick is built, add brick to built
20:         for child in brick.children do
21:             remove brick from child.parents
22:             if child.parents is empty then
23:                 add child to buildable
24:             end if
25:         end for
26:     end while
27: end function
```

Waiting area design

The waiting area for the supply station is design as an array of points evenly distributed on the concentric circle of the station (Fig. 4.9). This design makes all waiting drones have the same accessibility to the supply station, without relying on others. When a drone tries to resupply and the station is occupied, the server will allocate a nearest available position for waiting. The server then assigns an order to the drone. Following a first-in-first-out manner, the drone will go to resupply when all previous waiting drones have finished resupplying.

Flying altitude design

Illustrated in Fig. 4.10, a flight altitude is defined to make all drones navigate in the same height, which is also the idea behind the 2.5D world when modeling the problem. Restricting drones in the same height makes drones not affecting each other by thrust force. The flight altitude is designed to be high enough such that it is above all positions drones would reach. A waiting altitude is defined below the flight altitude for drones to wait at the supply station. It ensures the smooth switch for multiple drones between

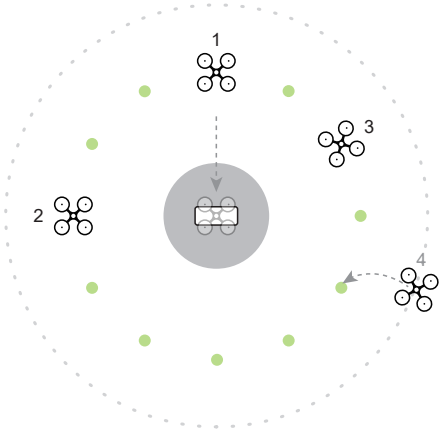


FIGURE 4.9: Waiting area for brick laying

states of navigating, waiting, and resupplying. Although introducing another altitude, the mechanism of the waiting area makes a drone at the waiting area being affected at a slight level by other drones.

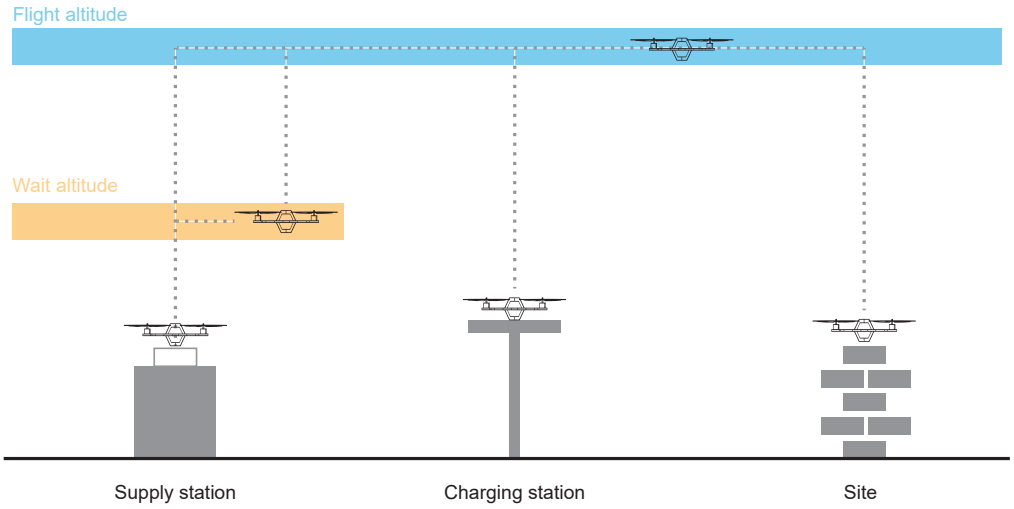


FIGURE 4.10: Flying altitude for brick laying

4.2.4 Results

Fig. 4.11 shows the building process of the bricklaying simulation. The experiment shows a robust collision-free approach and an obvious efficiency improvement over *The Flight Assembled Architecture Installation* by Augugliaro et al., 2014, which uses four drones to assemble a brick structure. The experiment also demonstrates the scalability and extensibility of the system for the smooth adaption of a various number of drones and different blueprints. Video is documented in Appendix B.

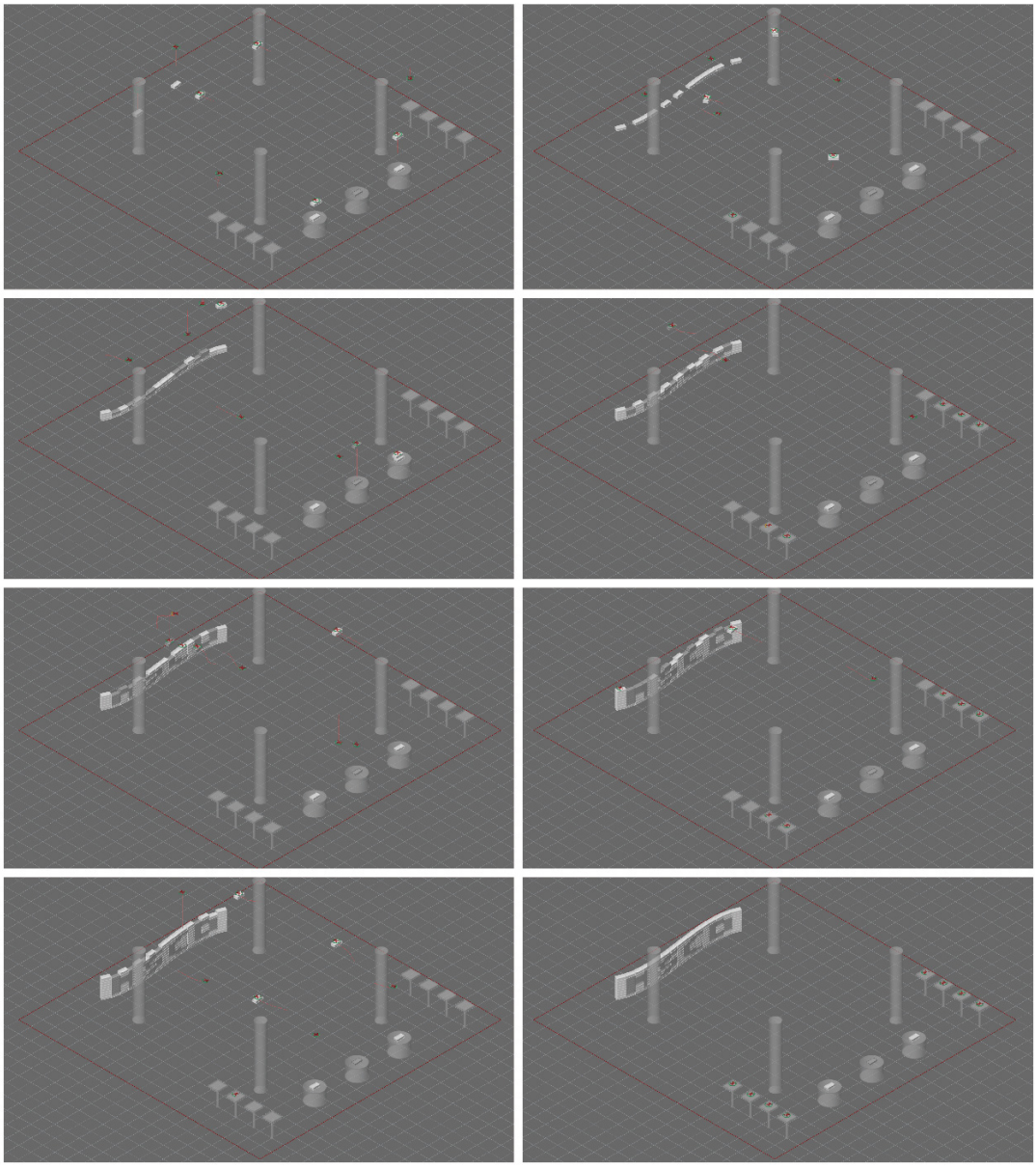


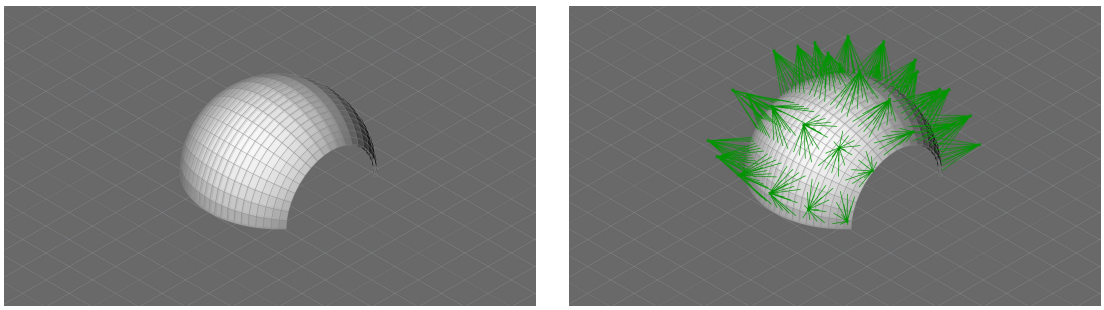
FIGURE 4.11: Screenshots of brick laying simulation process

4.3 Facade coating

Similar to the bricklaying experiment, this experiment uses the system to simulate the facade coating. The task is to let drones refill material at the supply station and spray at the coating position. Although the material is not discrete elements, since drones keep still when spraying, the process essentially resembles discrete element assembly. Waiting area and flight altitude design are identical to the previous experiment. However, the blueprint and build logics need a redesign.

4.3.1 Blueprint

Coating positions are defined as points on an offset surface of the original shell structure. Each coating position corresponds to a certain area on the original structure as shown in Fig. 4.12.



(A) Shell structure (B) Coating positions and coverage

FIGURE 4.12: Facade coating blueprint

4.3.2 Environment setup

As illustrated in Fig. 4.13, the environment consists of an uncoated structure, six charge stations, two supply stations, and four threat areas. The supply stations are homogeneous, drones are be assigned based on station occupancy or distance.

4.3.3 Facade coating implementation

The main loop for facade coating (Algorithm 5) is relatively simple compared to bricklaying. Coating positions defined in the blueprint is independent and sparsely distributed, the order doesn't matter and drone at one coating position won't affect drones at other positions. One coating position is assumed to be operated multiple times considering the limited material capacity of a drone.

4.3.4 Results

Fig. 4.14 shows the building process of the facade coating simulation. Similar to the previous experiment, the result shows the scalability and extensibility of the system. The experiment also demonstrates a bright future of using multiple drones to autonomously coat a building facade, which currently is still an unexplored field. Video is documented in Appendix B.

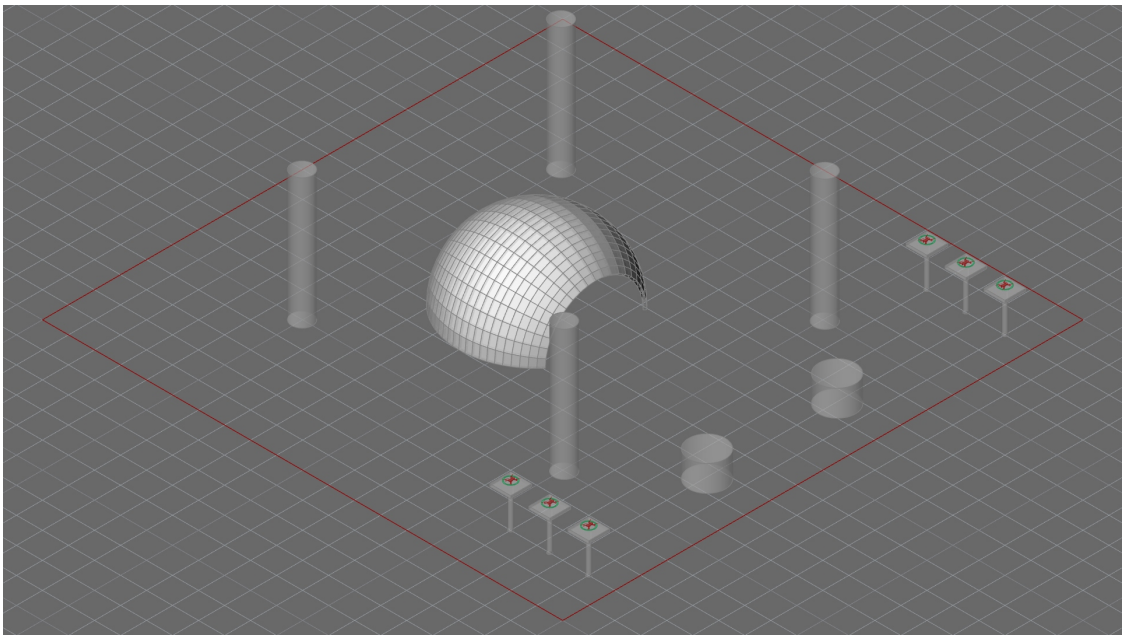


FIGURE 4.13: Environment setup for facade coating

Algorithm 5 Facade coating algorithm

```
initialize pos ← all coating positions
while pos is not empty do
    sample a position from pos
    assign a drone to coat facade at position
    if coating completes at position then
        remove position from pos
    end if
end while
```

4.4 Conclusion

The experiments demonstrate that the multi-drone autonomous construction system has the extensibility to be used in various scenarios. Changes in blueprints or entities can be directly reflected in real-time, which makes it easy to build different blueprints and deploy a different number of drones without modifying the system. Considering limited computing power, this thesis only tested a maximum number of 50 drones running at the same time for scalability, which is already enough for a middle-scale construction task and surpasses the number of drones in existing works by a great margin. However adding too many drones may cause problems like resource competing, a balanced number should be figured out in the real application. Moreover, this thesis only developed a minimum viable implementation for algorithms like navigation, refill, building, charging, and waiting, which is not necessarily robust enough for more complex scenarios.

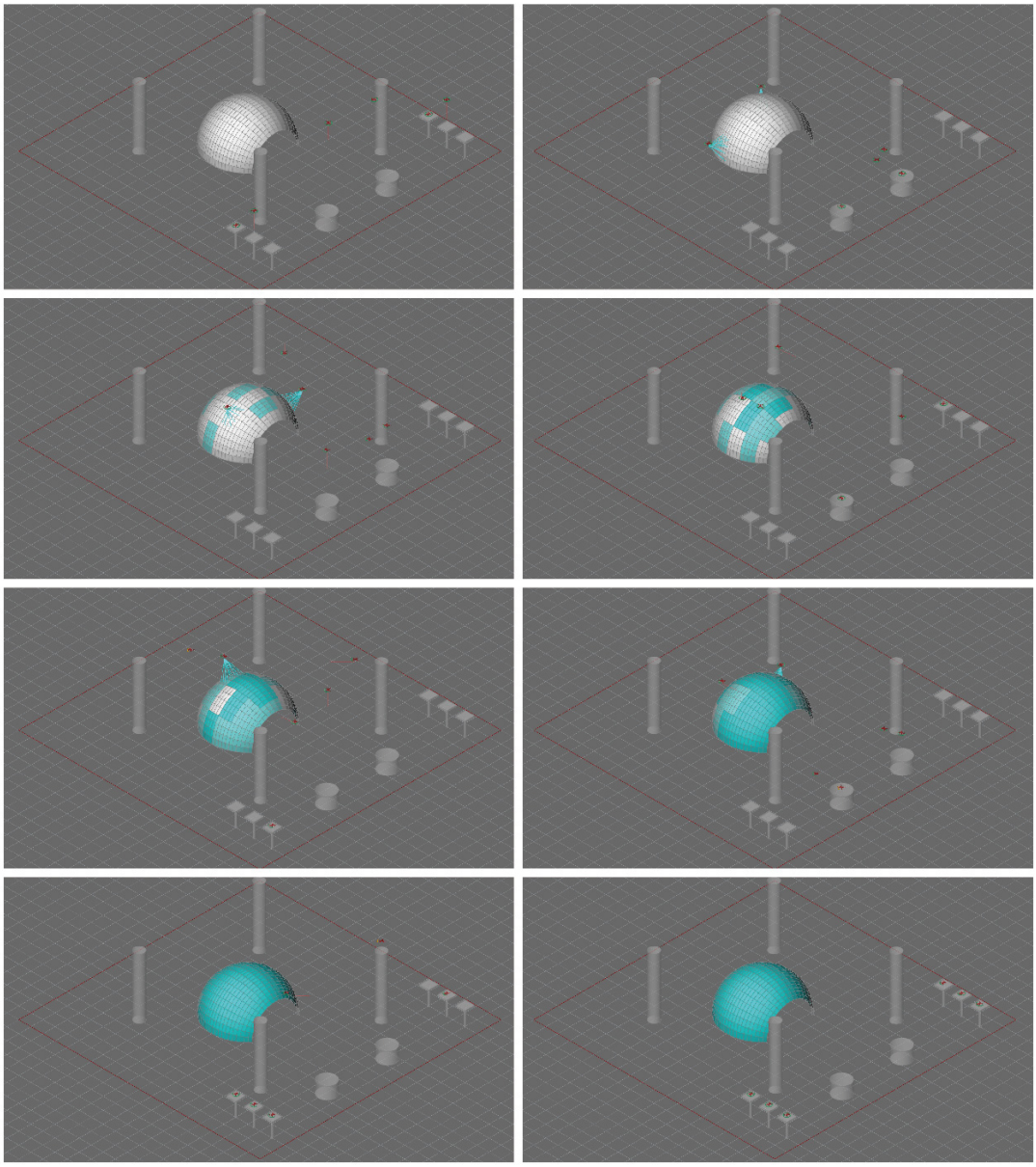


FIGURE 4.14: Screenshots of facade coating simulation process

Chapter 5

Conclusions

5.1 Conclusion

Aerial construction is a promising field of autonomous construction in the future with hardware development on load capacity and precision. However, there is no system developed for multi-drone autonomous construction in a dynamic environment and features scalability and extensibility. On the other hand, deep RL is making considerable achievements in robotics, but there is no work addressing multi-agent construction problems to the best of the author's knowledge. This work serves as a pilot study of building a multi-drone autonomous construction system by combining knowledge from multiple fields and further demonstrates the power of aerial construction.

Possible influences of this thesis include making designers experiment more easily in aerial construction, opening up a new possibility for drone research, paving way for a new paradigm shift in architecture design and construction. What's more, the general framework of the work can also be extended to some ground robot construction cases.

5.2 Contribution

Starting from introducing drone technology and autonomous construction, this thesis identifies the possibility of multiple drones actively participate in the construction process. After the literature review of previous research in drone-based construction and reinforcement, a gap is found in drone-based construction: the existing methodologies lack scalability and extensibility, which can be solved using deep reinforcement learning. Therefore, this thesis models the problem of multi-drone autonomous construction and builds a system to solve it. Simulation experiments demonstrate the viability of the system to address the gap.

The contribution is listed as follows:

- Model the multi-drone autonomous construction problem.
- Build a deep RL based system to solve multi-drone autonomous construction problem in dynamic environments that supports a scalable number of drones and entities and can be extended to various scenarios.
- Integrate the system with a design software along with some inspiring experiments for researchers' future use.

5.3 Limitations

This thesis aims to build a generic system for multi-drone construction. Although it seems promising for multiple drones to automate construction tasks with improved efficiency, limitations still exist. Precision and load capacity are two main hardware problems. The cost of development and deployment is also a problem that hinders the real application. What's more, drone compatible build materials and end effectors are under development fields. The limitations of the proposed system are listed below.

Problem formulation limitation

This technical part of the thesis mainly focuses on solving multi-drone collision-free coordination. Thus, the problem is abstracted neatly to solve this problem. However, the real application is always much more messy, sophisticated, and hard to model. For example, strong wind, scaffolds, material interlocking, human intervention, safety. By properly extending the problem modeling and system implementation by the case can address some problems, nevertheless, it is extremely hard to model and solve all possible situations in autonomous construction.

Collaboration limitation

In an asynchronous collaborative construction scenario, which this thesis mainly focuses on, drones complete tasks independently without relying on others. However, an additional algorithm is needed if controlling multiple robots to collaborate on a single task, like lifting a heavy object. Depending on tasks, robust rule-based algorithms, or multi-agent RL should be implemented.

Robustness limitation

Algorithm robustness should be more carefully examined when the number of participating drones grows up. For the navigation part, because the drones act according to a stochastic policy, it is necessary to add safe protection functions to avoid an accidental collision. Resource allocation is also a potential problem, a robust allocation algorithm should be developed. For other parts of the system, although rule-based algorithms are implemented to suffice the minimum requirement, some edge cases like battery failure, drastic environment forces, need to be robustly implemented in a real application.

Efficiency limitation

When the number of drones grows up, properly allocating resources is important for efficiency consideration. On one hand, resources like supply stations and charging stations should be allocated according to the number of drones. On the other hand, the resource allocation algorithm is an active research topic, which is not investigated in this thesis. Moreover, the efficiency of proposed algorithms should be examined, especially the computation burden for a drone onboard computer.

5.4 Future works

There are several directions that could potentially be explored in the future. First, real drone integration. This thesis is built upon an abstractly modeled simulation environment, gaps between real drone control are still identified. Second, the improvement of algorithms. Algorithms, especially macro-actions can be extended to perform more complex tasks. RL model can also be fine-tuned and integrated with some rule-based algorithms to ensure safety. Third, end effectors and construction material can be integrated as modules of the system. Finally, more construction scenarios are waited to be modeled and solved.

Appendix A

Code Usage

Code is available on GitHub: https://github.com/SakuraiSatoru/multi-drone-construction

A.1 Demo

Following is a simplest way to run a demo simulation.

1. Clone the fork of multi-agent-envs:
   git clone https://github.com/SakuraiSatoru/multiagent-particle-envs
2. Install environment:
   cd multiagent-particle-envs
   pip install -e .
3. Clone multi-drone-construction:
   git clone https://github.com/SakuraiSatoru/multi-drone-construction
4. Follow instructions in README.me if there is any update of usage.
5. Install all dependencies
6. Install software:
   cd multi-drone-construction pip install -e .
7. Open ./resources/multi-drone-construction.gh in Rhinoceros6 (Grasshopper)
8. Run driver script:
   python driver.py bricklaying
9. Connect Grasshopper by toggling Websocket battery

A.2 Code Structure

- **./multiagent-particle-envs/**: the simulation environment
- **./mdac/**: the multi-drone autonomous construction system
- **./vis/**: visualization related code and resources
- **./driver.py**: the system entry

Appendix B

Video

Showcase video is available at https://youtu.be/ijUTBdULz_0. Brick laying, facade coating and system dynamicity is documented.

Bibliography

Alonso-Mora, Javier et al. (2013). "Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots". In: *Distributed Autonomous Robotic Systems: The 10th International Symposium*. Ed. by Alcherio Martinoli et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 203–216. ISBN: 978-3-642-32723-0. DOI: 10.1007/978-3-642-32723-0_15.

Ammar, Mirjan (2016). "Aerial Construction: Robotic Fabrication of Tensile Structures with Flying Machines". en. PhD thesis. Zürich: ETH Zurich. DOI: 10.3929/ethz-a-010881983.

Augugliaro, F. et al. (2014). "The Flight Assembled Architecture installation: Cooperative construction with flying machines". In: *IEEE Control Systems Magazine* 34.4, pp. 46–64. ISSN: 1941-000X. DOI: 10.1109/MCS.2014.2320359.

Berg, Jur van den et al. (Apr. 2011). "Reciprocal n-Body Collision Avoidance". In: vol. 70, pp. 3–19. DOI: 10.1007/978-3-642-19457-3_1.

Bhatnagar, Shalabh et al. (2007). *Natural-Gradient Actor-Critic Algorithms*.

Braga, Rafael G, Roberto X. da Silva, and Alexandre C B Ramos (2016). "Development of a Swarming Algorithm Based on Reynolds Rules to control a group of multi-rotor UAVs using ROS". In:

Bryson, Mitch and Salah Sukkarieh (2007). "Building a Robust Implementation of Bearing-only Inertial SLAM for a UAV". In: *Journal of Field Robotics* 24.1-2, pp. 113–143.

Carmona, René, Mathieu Laurière, and Zongjun Tan (2019). *Model-Free Mean-Field Reinforcement Learning: Mean-Field MDP and Mean-Field Q-Learning*. arXiv: 1910.12802 [math.OC].

Cassandra, Anthony Rocco (1998). "Exact and Approximate Algorithms for Partially Observable Markov Decision Processes". AAI9830418. PhD thesis. USA. ISBN: 0591833220.

Coetzee, Louis (Sept. 2018). "Smart construction monitoring of dams with UAVs". In:

Foerster, Jakob et al. (2017). *Counterfactual Multi-Agent Policy Gradients*. arXiv: 1705.08926 [cs.AI].

Giftthaler, Markus et al. (2017). "Mobile robotic fabrication at 1:1 scale: the In situ Fabricator". In: *Construction Robotics* 1.1-4, pp. 3–14. ISSN: 2509-811X. DOI: 10.1007/s41693-017-0003-5. arXiv: 1701.03573.

Goessens, Sebastien, Caitlin Mueller, and Pierre Latteur (Oct. 2018). "Feasibility study for drone-based masonry construction of real-scale structures". In: *Automation in Construction* 94, pp. 458–480. DOI: 10.1016/j.autcon.2018.06.015.

Haarnoja, Tuomas et al. (2018). *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. arXiv: 1801.01290 [cs.LG].

Honig, Wolfgang et al. (Aug. 2018). "Trajectory Planning for Quadrotor Swarms". In: *Trans. Rob.* 34.4, 856–869. ISSN: 1552-3098. DOI: 10.1109/TRO.2018.2853613.

Hunt, G. et al. (2014). "3D printing with flying robots". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4493–4499. DOI: 10.1109/ICRA.2014.6907515.

Kaamin, Masiri et al. (Jan. 2018). "The Application Of Micro UAV In Construction Project". In: *Malaysian Construction Research Journal* 2.

Konda, Vijay and John Tsitsiklis (2000). "Actor-Critic Algorithms". In: *SIAM Journal on Control and Optimization*. MIT Press, pp. 1008–1014.

Kurak, Sevkuthan and Migdat I. Hodzic (2018). "Control and Estimation of a Quadcopter Dynamical Model". In: *Periodicals of Engineering and Natural Sciences (PEN)* 6, pp. 63–75.

Latteur, Pierre et al. (Aug. 2015). "Drone-Based Additive Manufacturing of Architectural Structures". In:

Lindsey, Quentin, Daniel Mellinger, and Vijay Kumar (2012). "Construction of cubic structures with quadrotor teams". In: *Robotics: Science and Systems* 7, pp. 177–184. ISSN: 2330765X. DOI: 10.7551/mitpress/9481.003.0028.

Liu, Yong et al. (2019). *Multi-Agent Game Abstraction via Graph Attention Neural Network*. arXiv: 1911.10715 [cs.AI].

Long, Pinxin et al. (2017). *Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning*. arXiv: 1709.10082 [cs.RO].

Lowe, Ryan et al. (2017). *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*. arXiv: 1706.02275 [cs.LG].

Mordatch, Igor and Pieter Abbeel (2017). "Emergence of Grounded Compositional Language in Multi-Agent Populations". In: *arXiv preprint arXiv:1703.04908*.

Park, S., L. Zhang, and S. Chakraborty (2017). "Battery assignment and scheduling for drone delivery businesses". In: *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6.

Pham, Huy Xuan et al. (2018). *Cooperative and Distributed Reinforcement Learning of Drones for Field Coverage*. arXiv: 1803.07250 [cs.RO].

Qie, H. et al. (2019). "Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning". In: *IEEE Access* 7, pp. 146264–146272.

Rashid, Tabish et al. (2018). *QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning*. arXiv: 1803.11485 [cs.LG].

Schulman, John, Sergey Levine, and Pieter Abbeel (2015). "Trust region policy optimization". In: *In ICML*, pp. 1889–1897.

Schulman, John et al. (2017). *Proximal Policy Optimization Algorithms*. arXiv: 1707.06347 [cs.LG].

Shamsoshoara, Alireza et al. (2018). *Distributed Cooperative Spectrum Sharing in UAV Networks Using Multi-Agent Reinforcement Learning*. arXiv: 1811.05053 [cs.MA].

Shin, M., J. Kim, and M. Levorato (2019). "Auction-Based Charging Scheduling With Deep Learning Framework for Multi-Drone Networks". In: *IEEE Transactions on Vehicular Technology* 68.5, pp. 4235–4248.

Silver, David et al. (2014). "Deterministic Policy Gradient Algorithms". In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, I–387–I–395.

Son, Kyunghwan et al. (2019). *QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning*. arXiv: 1905.05408 [cs.LG].

Sukhbaatar, Sainbayar, Arthur Szlam, and Rob Fergus (2016). *Learning Multiagent Communication with Backpropagation*. arXiv: 1605.07736 [cs.LG].

Sunehag, Peter et al. (2017). *Value-Decomposition Networks For Cooperative Multi-Agent Learning*. arXiv: 1706.05296 [cs.AI].

Tožička, Jan et al. (2019). "Application of Deep Reinforcement Learning to UAV Fleet Control". In: *Intelligent Systems and Applications*. Ed. by Kohei Arai, Supriya Kapoor, and Rahul Bhatia. Cham: Springer International Publishing, pp. 1169–1177.

Wang, Weixun et al. (2019). *From Few to More: Large-scale Dynamic Multiagent Curriculum Learning*. arXiv: 1909.02790 [cs.AI].

Wikipedia contributors (2020a). *Drone display — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/w/index.php?title=Drone_display&oldid=949242991.

— (2020b). *Unmanned aerial vehicle — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-March-2020]. URL: https://en.wikipedia.org/w/index.php?title=Unmanned_aerial_vehicle&oldid=946381220.

Willmann, Jan et al. (2012). "Aerial Robotic Construction towards a New Field of Architectural Research". In: *International Journal of Architectural Computing* 10.3, pp. 439–459. DOI: 10.1260/1478-0771.10.3.439. URL: https://doi.org/10.1260/1478-0771.10.3.439.

Wood, Dylan et al. (Sept. 2018). "Cyber Physical Macro Material as a UAV [re]Configurable Architectural System". In: pp. 320–335. ISBN: 978-3-319-92293-5. DOI: 10.1007/978-3-319-92294-2_25.

Yang, Bo and Min Liu (July 2018). "Keeping in Touch with Collaborative UAVs: A Deep Reinforcement Learning Approach". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, pp. 562–568. DOI: 10.24963/ijcai.2018/78. URL: https://doi.org/10.24963/ijcai.2018/78.

Yang, Yaodong et al. (2018). *Mean Field Multi-Agent Reinforcement Learning*. arXiv: 1802.05438 [cs.MA].

Zhang, Kaiqing, Zhuoran Yang, and Tamer Başar (2019). *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms*. arXiv: 1911.10635 [cs.LG].