Comparison of Team Robot Localization by Input Difference for Deep Neural Network Model

by

Sehyeok Kang

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2020 by the
Graduate Supervisory Committee:

Theodore P. Pavlic, Chair
Andréa W. Richa
Yezhou Yang

ARIZONA STATE UNIVERSITY

May 2020

ABSTRACT

In a multi-robot system, locating a team robot is an important issue. If robots can refer to the location of team robots based on information through passive action recognition without explicit communication, various advantages (e.g. improving security for military purposes) can be obtained. Specifically, when team robots follow the same motion rule based on information about adjacent robots, associations can be found between robot actions. If the association can be analyzed, this can be a clue to the remote robot. Using these clues, it is possible to infer remote robots which are outside of the sensor range.

In this paper, a multi-robot system is constructed using a combination of *Thymio* II robotic platforms and *Raspberry pi* controllers. Robots moving in chain-formation take action using motion rules based on information obtained through passive action recognition. To find associations between robots, a regression model is created using Deep Neural Network (DNN) and Long-Short-Term Memory (LSTM), one of state-of-art technologies.

The input data of the regression model is divided into *historical* data, which are consecutive positions of the robot, and *observed* data, which is information about the observed robot. *Historical* data is sequence data that is analyzed through the LSTM layer. The accuracy of the regression model designed using DNN can vary depending on the quantity and quality of the input. In this thesis, three different input situations are assumed for comparison. First, the amount of *observed* data is different, second, the type of *observed* data is different, and third, the history length is different. Comparative models are constructed for each case, and prediction accuracy is compared to analyze the effect of input data on the regression model. This exploration validates that these methods from deep learning can reduce the communication demands in coordinated motion of multi-robot systems.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Overview

Collaboration makes it efficient and effective to achieve tasks or goals that are difficult to accomplish alone. Likewise, in robot systems, it is often better for more robots to perform tasks together than for a robot to do tasks alone. For these reasons, various works about Multi-Robot System (MRS) (Khaldi and Cherif, 2015; Jevtić and Andina de la Fuente, 2007; Tan and Zheng, 2013) have been researched to collaborating the robots. Khaldi and Cherif (2015) emphasized that in order for robots to cooperate with each other, the one of the important factors is knowing the location of other robots. Also, according to Martinelli *et al.* (2005), knowing their position and orientation are essential in order to successfully doing their tasks. There are many ways for robots to get the information such as Peer-to-Peer communication, observation by sensors, and global communication.

However, there are many limitations and disadvantages to these explicit communications. For example, when they get their position from global communication, tremendous network communications are needed. Sometimes, a central commander is needed. This is also one of the factors that makes it difficult for the robot system to be decentralized. Also, sensors have trouble detecting the location of robots that are far away. To overcome these difficulties, methods using passive action recognition have been studied. Das *et al.* (2016) describes passive action recognition as "robots use sensors to directly observe the actions of their teammates". Choi *et al.* (2017) uses the passive action recognition. They have been conducted to refer non-local in-

formation from local information. They proposed that a robot could infer the remote robots' position by using nearby robots' position and they proved the feasibility of the idea through simulation.

Based on these possibilities, I expand an experiment to apply it to the real-world robotic system. In this experiment, the physical robots called as *Thymio* are used to set team robots. Moreover, the prediction model for the remote robot is implemented using Artificial Neural Network (ANN) technology. Deep Neural Network (DNN) and Long Short-Term Memory (LSTM), which Hochreiter and Schmidhuber (1997) suggested, are applied to infer the position of remote robots.

Finally, the accuracy of the model may vary depending on the information used to predict the remote robot. To compare this, several models with different inputs for inference are implemented. The comparative models are divided into three cases. First, when the number of observed robots is different, second, when only a part of the information of the observed robot is used, and third, when the history length is different. In each case, the difference in accuracy between the control model and the comparison model is compared and analyzed.

1.2 Motivations

One of the most recent noticeable technologies in multi-robot system is Swarm Robotics (SR). Researchers have come up with ideas from nature for robots to collaborate efficiently. There are many swarms in nature. In a swarm, individuals follow only simple rules, but these actions come together to achieve a specific goal. For example, in the case of foraging by some species of ants, the ants follow two simple rules. First, the ants find the prey and lay pheromones on their way home. Second, the ant travels in a path where the pheromone concentrations are stronger. As a result, the pheromone is scattered heavily in the path where the ants travel frequently, which

is the shortest path because ants can best reinforce the trail before it evaporates. Swarm Intelligence (SI) is made up of applicable algorithms by analyzing behaviors in nature, and SR is a robot system using SI. SR generally does not use global information. This is because there is no central commander to collect and deliver global information and robots configured by SR take action by simple rules based on local information.

However, if robots in SR can deduce global information (non-local information) using limited information (local information), they may be able to achieve their goals more efficiently. For example, suppose some smart robots are in a swarm. These smart robots are able to catch global information and use it to steer the entire group to the desired direction. An example is the caging scenario in Choi *et al.* (2017). They suggested that if the robot at the rear of team-robots going in a row can predict the location of the lead robot, the rear robot (Tail) can move to speed up the pace of the team surrounding the target.

To infer global information using local information in SR, the following process is required. Each robot observes behavior or movement changes of adjacent robots and reflects it in its own actions in SR. Thus, changes in adjacent robots affect its behavior and there is an association between them. Biologists and behavioral ecologists call clues to analyze these associations as 'signal' or 'cue'. For example, in Das *et al.* (2016), they have mentioned that there are patterns in the waggle dance of bees, and other bees can identify the patterns by analyzing the cues of the waggle dance. They also implemented a multi-robot system using these cues and studied how to transmit information through passive communication.

In this experiment, the behavior of robots affects the behaviors of adjacent robots. Therefore, there will be an association in the behavior between two adjacent robots, which can be used as a cue to analyze the association. Thus, if we can analyze this

association using the cues, we can predict the movement of unobserved individuals further away. If these predictions are repeated recursively, the robot will be able to predict the movement of the whole robots by observing only the movement of the nearest robot.

To prove this, this study implements the robots in chain formation by simple rules using the principle of SR. In fact, the experiment is not a complete SR implementation. SR is based on homogeneous robots, but in this experiment, the robots are heterogeneous because the head and tail robot move with a slightly different motion rule than other robots (follower robots). However, since followers and tail robots operate using only local information, it can be said that decentralization and local sensing, which are important characteristics of SR, have been applied.

In addition, the technique used to make predictions in the experiment is Deep Neural Network (DNN) technique, one of the state-of-art techniques. From the past to the present, many models have been proposed to improve the accuracy of DNN. One of them is the Long Short-Term Memory (LSTM) model proposed by Hochreiter and Schmidhuber (1997). LSTM is one of the Recurrent Neural Network models and is very effective for analyzing time series data. The LSTM layers are added to the DNN prediction model to improve accuracy in order to analyze the continuous movement of adjacent robots. Because models implemented in the experiment use robots' continuous movement, it exists as sequence data. Therefore, an LSTM layer that is easy to analyze sequence data is used.

1.3 Challenges

There are two challenges in the experiment. First, implementing a team-robot on a real-world robot platform. This experiment is based on the team-robot of Choi *et al.* (2017). However, The experiment of Choi *et al.* (2017) was simulated using

a multi-robot simulator only and not conducted using a real robot. Therefore, the experiment should build a real robot platform and implement it so that robots move in accordance with the rules on the platform. In particular, simulation results and experimental results in the real world may have large gaps.

In Ligot and Birattari (2019), a real robot swarm was studied to reduce the gap between simulation-only experiment and reality. Similarly, in the experiment, it is important to construct a robot platform to reduce the gap between the simulation results in Choi *et al.* (2017) and the reality experiment. The role of robots in the platform is played by *Thymio* and *Raspberry pi*. *Thymio* is a robot made for educational purpose and can be operated using various OS when used with *Raspberry pi*. One drawback to *Thymio* is that the sensor range is too short, so it is not suitable for experimenting. In order to solve this problem, the central computer and camera function as high performance sensors as are commonly used in robot experiments. Specifically, the central computer is designed to convert the position and direction of the robots captured from the camera to the relative coordinate and deliver only the corresponding local information for each robot. *'open-cv'*, a vision package for *python library*, is used to calculate the robots' position and orientation in images captured in the video.

Next, in our inferences design, we must increase the accuracy of predictions about remote robots. When a robot makes pose based on local information, the number of possible poses of remote robots is several, not one. Therefore, due to the large number of predictable cases, the accuracy of the prediction is inevitably low if the DNN model is simply configured. To address this, the predictive model should be given more clues to make more accurate predictions. The experiment assumes that the robots know global information before r time steps. Sequential position for robots before r time steps is provided as a clue for the predictive model to make more accurate predictions.

The predictive model uses the LSTM layers to use the sequential data for training and predicting.

1.4 Contributions

If robots can infer global information using only local information, they have various advantages. For instance, if information can be obtained through passive communication without using explicit communication, the amount of communication can be reduced. Considering the use of multi-robots for military purposes, reducing the amount of communication can be a significant merit. This can reduce the risk of being detected by the enemy or exposing information to the enemy due to communication. This is a significant merit in terms of security. Also, central commander is generally needed to use global communication. However, in the case of global communication with the central commander, it will be difficult for the multi-robot system to be decentralized. Therefore, this experiment will help to implement decentralized robot system.

1.5 Outlines

Chapter 2 discusses the previous work and related work to help understand the experiment. Chapter 3 explains the method used for comparison. The control model used as a control model is described, and the comparison models according to each comparison method are described. Chapters 4 and 5 describe the method for experimentation. The method used to implement the physical robot platform is described, and the collected data through robots operating on the platform is described. Chapter 6 analyzes the prediction results by comparison model and mentions the difference in accuracy by model. Chapter 7 discusses conclusions and future work.

Chapter 2

BACKGROUND AND RELATED WORKS

Swarm Robotics (SR) is a subdivision of collective robotics, one of the types of Multi-Robot Systems (MRS), in which robots act according to simple rules and cooperate with robots. In the experiment, robots move in chain formation using the principle of SR. Also, in the experiment, the robots determine their own behavior using only local information without a central commander. These characteristic is similar to that of SR. Therefore, it is advantageous to have knowledge of SR to help understand the experiment. Also, in the experiment, Artificial Neural Network (ANN) is introduced, one of the latest technologies, to infer global information. Thus, understanding the SR and basic knowledge of ANN is necessary to understand this thesis. This chapter gives literature reviews of MRS, SR and ANN technologies. Moreover, the previous researches (Choi *et al.*, 2017, 2020) which have been conducted before the experiment are introduced to help understand the experiment.

2.1 Multi-Robot System (MRS) and Swarm Robotics (SR)

Some tasks can be too complex and difficult to perform with a single robot. Also, when several robots work together, a task can be done faster and more effectively than a single robot by itself. Multi-Robot System (MRS) is a system designed to allow multiple robots to work together and perform tasks harmoniously. Khaldi and Cherif (2015) introduced MRS as a method designed to overcome the difficulties of a single robot and mentioned that MRS has many advantages over performing missions with only a single robot. Khaldi and Cherif (2015), Jevtić and Andina de la Fuente (2007), and Tan and Zheng (2013) used MRS to perform various works such

as collective robotics and distributed robotics.

However, there are some disadvantages to MRS. Khaldi and Cherif (2015) mentioned that MRS faces a decentralized problem. In general, MRS has a centralized control system and robots in MRS act based on information from the central commander. Thus, the central commander controls whole robots. Accordingly, MRS is difficult to build a decentralized system. Swarm Robotics (SR) is the method that emerged to build decentralized system in multi-robot system. Dorigo *et al.* (2004) first mentioned SR in the "swarm-bot" project. SR works on the principle of Swarm Intelligence (SI), which is an algorithm inspired by swarms in nature. In nature, there are many cases such as ants, bacterial colonies, or fish schools that live in harmony in swarms. In the swarms, each member does not determine its next action based on information about whole (global) but rather the next action based on information available from adjacent members' behavior. Many algorithms, such as an Ant Colony Optimization (ACO), Particle Swarm Optimization Algorithm (PSO), Artificial Fish Swarm Algorithm (AFSA), have been created that analyze the behavior in nature and apply it similarly to solve problems. These algorithms are called SI. The characteristics of SR compared to MRS can be seen in the Table 2.1. According to Tan and Zheng (2013), SR has the following characteristics.

	Swarm Robotics	**Multi-Robot System**
Population size	Variation	Small
Control	Decentralized / Autonomous	Centralized / Remote
Homogeneity	Homogenous	Heterogeneous
Flexibility	High	Low
Scalability	High	Low

Table 2.1: Characteristic Comparison of SR and MRS From Khaldi and Cherif (2015)

1. Local sensing and communication : In SR, robots are assumed to have limited sensor range and limited communication capabilities. This is because, even in swarm in nature, each entity knows only the information of its neighbor. Thus, there is no central commander in SR and the range of sensor is limited, so global information cannot be obtained. Swarm robots therefore operate using only limited information.

2. Autonomous and Decentralized : Central command does not exist in SR and global information is not used. When using global information, flexibility and scalability may be limited. However, swarm robots use only simple rule based on local information. As a result, the robots are decentralized and autonomous.

3. Large number and Flexibility : The number of roles of robots in swarms should be as small as possible. Swarm robots are usually designed to do the same thing. Thus, swarm robots have homogeneous characteristics. Also, since robots are not controlled by a central commander, it is not difficult to increase or decrease the number of robots. Therefore, SR can effectively operate a large number of robots, and the system has high flexibility.

The experiment uses the characteristics of SR. In the experiment, the robots are decentralized and moved without the command of the central commander. In addition, all robots except head robot are operated by the same rule and they obtain information through observation only within the limited sensor range and use the information. The same rule is proposed by the Choi *et al.* (2017), and the robots move in chain formation using the rule.

Nouyan *et al.* (2008) and Maxim *et al.* (2009) have also studied how swarm robots move in chain formation. The difference from this paper, however, is that this experiment is about obtaining global information using only local information, when robots move in chain formation by the motion rule. Existing studies have aimed at forming

formations, while this experiment focuses on inferring the position of remote robots.

2.2 Artificial Neural Network (ANN) and Long-Short-Term Memory (LSTM)

The basic concept of Artificial Neural Network (ANN) is based on how the human brain solves problems. A perceptron, a component of ANN, is built on the principles of neurons in the human brain. A Multilayer Perceptron (MLP) is a model in which several perceptron form one layer and several layers exist. MLP consists of input layer, hidden layer, and output layer. The input layer is the first layer in MLP and is a layer into which data for model training or prediction is input. The output layer is the layer that outputs the results. Layers between the input and output layers are called hidden layers.

Network models that have more than one hidden layer are called Deep Neural Network (DNN). DNN can solve problems that a single perceptron or other machine learning models, such as Support Vector Machine (SVM) or Logistic Regression, cannot solve. For instance, a DNN provides solutions to nonlinear problems such as the XOR problem. Given enough data to use for training, DNN can be an effective model. Therefore, in this experiment, the prediction model is conducted using DNN. However, the basic DNN model is not effective for training with sequential data. Data with sequences are not independent of each other because they have a specific order.

However, the general model assumes that all input data are independent and identically distributed. Recurrent Neural Network (RNN) has been proposed for effective training using sequential data Raschka and Mirjalili (2019). In a general feed-forward network model, data propagates in one direction from the input layer to the output layer. RNN, however, output from the hidden layer is delivered to the hidden layer of the next time step. Training in the next time step proceeds considering the training result in the previous time step. Therefore, RNN is effective for training

using sequential data. However, when the length of the sequences becomes longer, Gradient Exploding or Gradient Vanishing problem occurs.

The model that solved this problem is Long Short-Term Memory (LSTM). LSTM was first proposed in Hochreiter and Schmidhuber (1997). The LSTM is advantageous when learning using data with long sequences. In RNN, Gradient Vanishing or Exploding problems occur when the weight (w) is greater than or less than 1. Since the weights are continuously multiplied as the learning progresses, if the weight is not maintained at 1, the value disappears or increases rapidly. In the LSTM, a cell state is added from the RNN. The cell state classifies data into long-term memory and short-term memory. The memory cell keeps the weight (w) to 1 by using long-term and short-term memory appropriately. By doing this, it can solve these problems. In this experiment, the positions of the robots have a sequence. To utilize the sequence data, the LSTM layer is used to predict the next position.

2.3 Previous Work : Remote Team Robot Localization

Research has been conducted to find the location of robots in a multi-robot system. In particular, in case of swarm robots, the robots use only local information to make a decision, and so it is important to accurately determine the location of the robot. Jiang *et al.* (2007) and Kelly and Martinoli (2004) have been researching methods for teammate localization using light or infrared sensors. Research on teammate localization have been conducted a lot, but research on inferring the location of global teammates using local information has rarely been conducted.

Choi *et al.* (2017) first proposed a method of inferring global information using local information in team-robot. They suggested a method of inferring remote robot's position and orientation based on local information when robots except head robot move in chain-formation using the same motion rule. They made a regression model

by using DNN for prediction and tested the model using a team-robot simulation program called 'Robotarium'.

In addition, Choi *et al.* (2020) carried out experiments to the actual robot frame. Choi *et al.* (2020) adds LSTM layers to the model, for more accurate results. The experiments presented in this thesis are extensions of Choi *et al.* (2017) and Choi *et al.* (2020) experiments, and the chain formation is formed with real robots using the motion rule proposed by Choi *et al.* (2017).

In this thesis, the principle of how team-robots work, and the physical frame are described in more detail. Moreover, the number of robots in team-robots is increased to seven, and the difference in prediction results are compared according to the difference in the amount of input data in the prediction model.

Chapter 3

METHOD

3.1 Problem Definition

One of the characteristics of swarm robot is a local sensing and local communication. The robots in the swarm do their behavior or judgement using the information detected within the limited sensor range and limited communication. Therefore, the behavior of a robot affects the behaviors of adjacent robots. Particularly, when the robots move in line, robots may be directly affected by the movement of the robots ahead and behind them. If the association between the behaviors of adjacent robots can be analyzed, the movement of the robot outside the sensor range may be predictable. The way to communicate through the association between actions is called behavioral communication. In Novitzky *et al.* (2012) and Das *et al.* (2016), the "Bee waggle dance" (Dutta *et al.*, 2012) was applied to the robot to implement behavioral communication. In this experiment, the behavior of neighboring robots is analyzed to find the association and the potential clues of the behavior are used to predict the remote robot.

In this thesis, behavioral association is learned using Deep Learning technology which is one of the state-of-the art technologies. The accuracy of the model will vary depending on the input used for analysis. Therefore, in this experiment, three methods are used to compare the results depending on the input.

In the first case, I study how the number of sensed robots affects prediction performance. The amount of local information to be obtained can vary depending on the range of sensor radius. The sensor performance of the robot is assumed to be the

number of robots that can be observed in chain formation. In this experiment, the accuracy of the prediction may be changed according to the number of cues. Assuming that the sensor range of the robot is different, three cases in which the number of cues is different are considered. Through this, I compare the result of three cases and analyze the accuracy difference according to the number of cues.

In the second case, I study how the kind of cue information affects prediction. Even when the three robots are observable, the results of using some of the information of the three robots or using the entire information may be different. All three robots' data may be helpful for prediction, but on the contrary, some may hinder accurate prediction. Two models with different cues used as input data are proposed, and the difference of the prediction results according to the types of cues is analyzed using them.

In the third case, I study the effect of history length on prediction. This experiment uses the model proposed by Choi *et al.* (2020) as the control model. The control model uses global information as history data. History data is concatenated with observed data through the LSTM layer. Depending on the history length, the LSTM layer may use more significant values for prediction, or vice versa. As a result, the result of the prediction may vary. Therefore, I propose models that have different history lengths and analyze the effects of history length on the results. In addition, this experiment uses scalable prediction approach, which uses the predicted results for the prediction of other robots for further prediction. This allows predictions for robots that are farther away, as well as predictions for the nearest robot that is outside the sensor range.

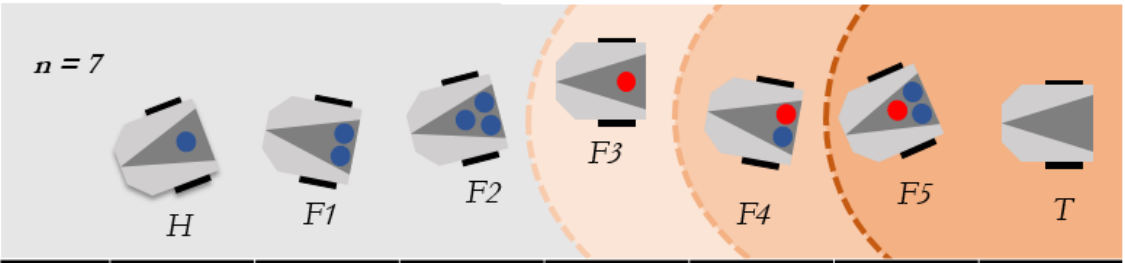
3.2 Theorem for Notation of Robot and Terminology

3.2.1 Notation of Robots in Chain-formation

In this experiment, the robots are not distributed and are lined up in a row. Therefore, there is an order from the front to the rear. When there is a total of n robots (# of robots=n) and they are numbered in order from the front robot, the front robot ($i = 1$) is called the head robot (H). The rearmost robot ($i = n$) is called Tail (T). Robots between H and T ($i \in \{2, \cdots, n-1\}$) are called Follower. Followers are numbered in ascending order from H closest such as $F_1, F_2, \cdots, F_{n-2}$ When robots are lined up as shown in the Fig. 3.1, the names of each robot are as shown in the Fig. 3.1.

3.2.2 Notation of Robots According to the Sensor Range

The experiment is carried out with a different sensor range of the robot, so the robot is given a name according to the sensor range. The range of the sensor is indicated by R. The meaning of $R = k$ is that the sensor can observe k nearest robots. In the experiment, the standard robot, which observes other robots and predicts remote robot's position and orientation, is T because all predictions are made by T. For example, if $R = 1$, T can only observe F_5 in the Fig. 3.1. Observable robots are represented by O_x according to the range of observations. The robot is marked O_1,O_2,O_3 in the order close to T. In addition, the robots which are out of the observation range, that is, the robot to be predicted, is denoted by P_x. Similarly, P_x is named as P_1,P_2,$\cdots$,P_k in the order close to T. The names of the robots according to R are shown in the Fig. 3.1.



Range	H	F1	F2	F3	F4	F5	T
R=1	P_5	P_4	P_3	P_2	P_1	O_1	O_0
R=2	P_4	P_3	P_2	P_1	O_2	O_1	O_0
R=3	P_3	P_2	P_1	O_3	O_2	O_1	O_0

Figure 3.1: Examples of Robot Notation According to the Absolute Position & The Notation According to the Sensor Range(R)

3.2.3 Position and Orientation Terminology

The information of the robot used in the experiment is divided into position and orientation. Position is represented by the (x, y) coordinate value on the two-dimensional plane, and orientation is represented by the degree value. Therefore, position is represented as $\overrightarrow{P}^t_R = (x^t_R, y^t_R)$. The superscript t represents the time step and the subscript R represents the corresponding robot. Therefore, $\overrightarrow{P}^t_H$ means the position of the H robot at time step t. Similarly, θ^t_H refers to the orientation of the H robot at time step t.

3.3 Control Model (Model 1)

In the experiment, the predictive model for use as a comparison is from Choi *et al.* (2020). In that work, a model for predicting a robot that is not observed using the information of one adjacent robot (1 cue) is used. This model is called 'Model 1 ($M1$)'. Deep Learning Technology is used to predict. The structure of the Model 1 is the Fig. 3.2.

The model uses both fully connected layers and LSTM layers. The input data is

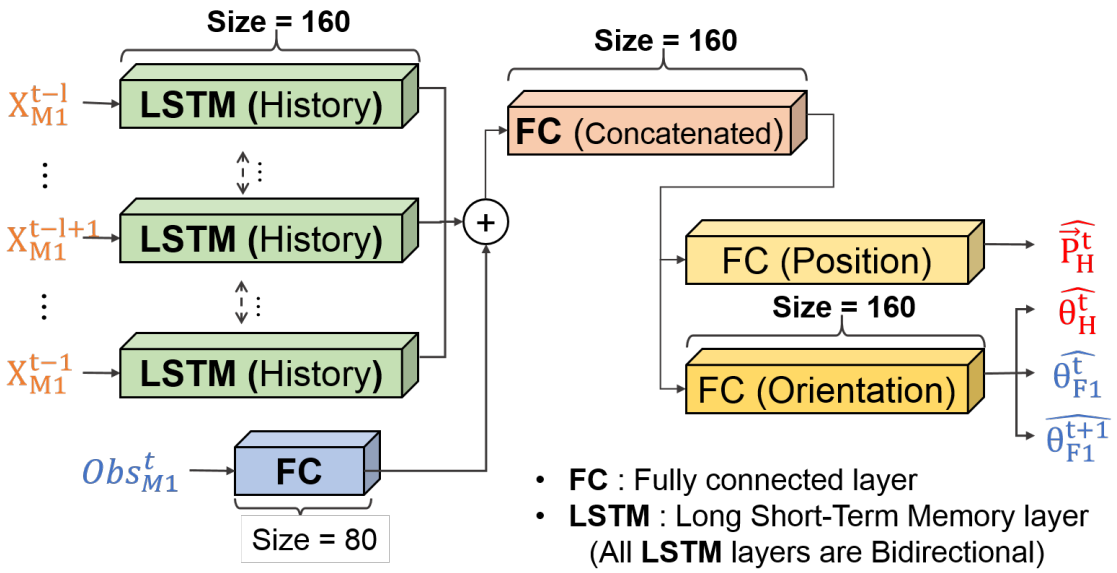


Figure 3.2: A Structure of the Control Model From Choi *et al.* (2020)

divided into *historical* data and *observed* data depending on the type. The model assumes that robots know global information (whole robot's position and orientation) from $t-r$ to $t-1$ time steps. *historical* data consists of global information. Therefore, *historical* data($Hist$) is composed of the position and orientation data of the robots up to $t-r$ to $t-1$ time steps.

$$Hist_{M1}^{t-r\to t-1} = (X_{M1}^{t-r}, X_{M1}^{t-r+1}, \cdots, X_{M1}^{t-1}) \quad , \quad X_{M1}^{t} = (\overrightarrow{P}_{H}^{t}, \theta_{H}^{t}, \overrightarrow{P}_{F_1}^{t}, \theta_{F_1}^{t})$$

Since Choi *et al.* (2020) assumes that information up to 10 time steps is known, it consists of global information from $t-10$ to $t-1$ time step data($Hist_{t-10\to t-1}^{M1}$). Since *historical* data is data related to the continuous movement of the robot, it uses LSTM layers which is effective for time series analysis. The data passes through the LSTM layers and then combined with one fully connected layer. *Observed* data(Obs) is information on adjacent robots observed by the robot in t and $t+1$ time steps. Unlike *historical* data, *observed* data only configures the information of robots observed. Among the information of the robots, the relative position can be calculated, but it

is believed that there is a limitation to accurately measure the orientations of the robots. Thus, *observed* data uses only the position of the robots.

$$Obs_{M1}^{t} = (\overrightarrow{P}_{F_1}^{t}, \overrightarrow{P}_{F_1}^{t+1})$$

The data passes through hidden layers and configured one fully connected layer. The two fully connected layers are concatenated together and reconstructed to one layer. This layer is separated into layers that predict positions and orientations. The model predicts $\overrightarrow{\tilde{P}}_{H}^{t}, \tilde{\theta_H^t}, \tilde{\theta_{F_1}^t}, and \;\; \tilde{\theta_{F_1}^{t+1}}$. The labels of this neural network model are divided into position ($\overrightarrow{\tilde{P}}_{H}^{t}$) and orientations ($\tilde{\theta_H^t}, \tilde{\theta_{F_1}^t}, and \;\; \tilde{\theta_{F_1}^{t+1}}$) and each label applies to different loss function. The values indicating the position ($\overrightarrow{\tilde{P}}_{H}^{t}$) is (x_H^t, y_H^t) in 2-dimensional space. The Loss function for position uses Euclidean distance. Euclidean distance is used to minimize the distance between the predicted coordinate $(\tilde{x}_H^t, \tilde{y}_H^t)$ and the real coordinate (x_H^t, y_H^t).

$$Loss_1 = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\tilde{x_i}_H^t - {x_i}_H^t)^2 + (\tilde{y_i}_H^t - {y_i}_H^t)^2}$$

In case of orientation, the Mean Squared Error(MSE) is used as a loss function.

$$Loss_2 = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\tilde{\theta_i} - \theta_i)^2}, \quad \theta \in \{\theta_H^t, \theta_{F_1}^t, \theta_{F_1}^{t+1}\}$$

The MSE proceeds to training the model to reduce the difference between the predicted value $\tilde{y}$ and the label y. Also the Gradient Vanishing problem is solved using a *ReLU* activation. The *ReLU* activation works between all layers except the last layer. The model can infer the remote robot's position and orientation using one adjacent robot's information (1 cue) Also, it is used to predict farther robots based on the information of the predicted robot. This model is used for comparison with the models to be presented in the next section.

3.4 Comparison 1 : Different Number of Cues

The type and amount of local information that can be obtained according to the range of the robot's sensor may be different. In the Fig. 3.1 , Ellipses represent the sensor ranges. If $R = 1$, T can only observe F_5, the nearest robot. Therefore, there is only one cue available. However, when $R = 2$ or $R = 3$, since T can observe F_4 and F_5 or F_3,F_4, and F_5, the number of available cues increases. In practice, if the robots are overlapping within the range of the sensor, it can be difficult to obtain accurate data by observation only. However, it has been able to overcome with other methods and many experiments have been conducted to solve this problem. Therefore, in this experiment, overlapping problems are excluded.

Depending on the amount of information to be used in prediction, the accuracy of the prediction will vary. Two models with different number of cues used for prediction are proposed. The structure of these models is similar to the control model, but the number of cues entered are different. The models are used to analyze the differences in the results according to the input information.

3.4.1 Model2(M2) : Use 2 Cues

A model assumes that the robot is observable up to two adjacent robots. $(R = 2)$ Therefore, this model uses two cues to predict the position and orientation of the third remote robot. Information used as input data is divided into $Hist$ and Obs, as is the control model. The difference from the control model is that the $Hist$ data of the model consists of information about four robots. Since the Obs data consists of the observed information, positions of two robots are included. The $Hist$ data and Obs data of this model are as follow.

$$Hist_{M2}^{t-r\rightarrow t-1} = (X_{M2}^{t-r}, X_{M2}^{t-r+1}, \cdots, X_{M2}^{t-1}) \quad , \quad X_{M2}^{t} = (\overrightarrow{P}^{t}_{H}, \theta^{t}_{H}, \overrightarrow{P}^{t}_{F_1}, \theta^{t}_{F_1}, \overrightarrow{P}^{t}_{F_2}, \theta^{t}_{F_2})$$

$$Obs^t_{M2} = (\overrightarrow{P}^t_{F_1}, \overrightarrow{P}^{t+1}_{F_1}, \overrightarrow{P}^{t+1}_{F_2}, \overrightarrow{P}^{t+2}_{F_2})$$

Input data configured like this pass LSTM layers and FC layers respectively and then predict $\overrightarrow{\tilde{P}}^t_H, \tilde{\theta^t_H}, \tilde{\theta^t_{F_1}}, \tilde{\theta^t_{F_2}}$, and $\tilde{\theta^{t+1}_{F_2}}$. Moreover, Model2 is designed with the assumption that the clue which helps to predict target robot propagates over time steps. If the clue for prediction is moved through adjacent robots and relay propagated, it will propagate to the next robot over time. For example, in the figure, when $R = 2$, the robots that can be observed are $F_4(O_1)$ and $F_5(O_2)$. $\overrightarrow{P}^t_{F_3}(P_1)$ affect $\overrightarrow{P}^t_{F_4}(O_1)$ and $\overrightarrow{P}^{t+1}_{F_4}$. They affect $\overrightarrow{P}^{t+1}_{F_5}(O_2)$ and $\overrightarrow{P}^{t+2}_{F_5}$. Becasue $\overrightarrow{P}_{F_5}(O_2)$ is not directly affected by the behavior of $F_3(P_1)$, but by the behavior of $F_4(O_1)$, $\overrightarrow{P}^t_{F_4}(O_1)$ and $\overrightarrow{P}^{t+1}_{F_4}$ affects $\overrightarrow{P}^{t+1}_{F_5}(O_2)$ and $\overrightarrow{P}^{t+2}_{F_5}$. That is, $\overrightarrow{P}^t_{P_1}$ affects $\overrightarrow{P}^{t+i}_{O_{(R-i)}}$ and $\overrightarrow{P}^{t+i+1}_{O_{(R-i)}}$. Thus, in the $M2$, the Obs is given as input with data of different time steps.

3.4.2 Model3(M3) : Use 3 Cues

This model assumes that the robot has a wider range sensor than that of Model 1 and Model 2. ($R = 3$) The model uses information from three adjacent robots to predict information about the fourth-most remote robot. Similar to the Model 2,

	$Hist^{t-r\to t-1}_{Mx} = (X^{t-r}_{Mx}, X^{t-r+1}_{Mx}, \cdots, X^{t-1}_{Mx})$	Obs^t_{Mx}	**Labels**
M1	$X^t_{M1} = (\overrightarrow{P}^t_H, \theta^t_H, \overrightarrow{P}^t_{F_1}, \theta^t_{F_1})$	$\overrightarrow{P}^t_{F_1}, \overrightarrow{P}^{t+1}_{F_1}$	$\overrightarrow{\tilde{P}}^t_H, \tilde{\theta^t_H}$, $\tilde{\theta^t_{F_1}}, \tilde{\theta^{t+1}_{F_1}}$.
M2	$X^t_{M2} = (\overrightarrow{P}^t_H, \theta^t_H, \overrightarrow{P}^t_{F_1}, \theta^t_{F_1}, \overrightarrow{P}^t_{F_2}, \theta^t_{F_2})$	$\overrightarrow{P}^t_{F_1}, \overrightarrow{P}^{t+1}_{F_1}$, $\overrightarrow{P}^{t+1}_{F_2}, \overrightarrow{P}^{t+2}_{F_2}$	$\overrightarrow{\tilde{P}}^t_H, \tilde{\theta^t_H}$, $\tilde{\theta^t_{F_1}}, \tilde{\theta^t_{F_2}}, \tilde{\theta^{t+1}_{F_2}}$.
M3	$X^t_{M3} = (\overrightarrow{P}^t_H, \theta^t_H, \overrightarrow{P}^t_{F_1}, \theta^t_{F_1}, \overrightarrow{P}^t_{F_2}, \theta^t_{F_2}, \overrightarrow{P}^t_{F_3}, \theta^t_{F_3})$	$\overrightarrow{P}^t_{F_1}, \overrightarrow{P}^{t+1}_{F_1}$, $\overrightarrow{P}^{t+1}_{F_2}, \overrightarrow{P}^{t+2}_{F_2}$, $\overrightarrow{P}^{t+2}_{F_3}, \overrightarrow{P}^{t+3}_{F_3}$	$\overrightarrow{\tilde{P}}^t_H, \tilde{\theta^t_H}$, $\tilde{\theta^t_{F_1}}, \tilde{\theta^t_{F_2}}$, $\tilde{\theta^t_{F_3}}, \tilde{\theta^{t+1}_{F_3}}$.

Table 3.1: Hist,Obs,Labels According to the Model(M1,M2,M3)

Model 3 assumes that clues propagate over time steps. Thus, the observed data of model 3 consists of data with different time steps. The difference between the model 1, 2, and 3 is the difference in the amount of input data. I compare the three models to verify that differences in input data lead to differences in accuracy. Table. 3.1 lists the $Hist$, Obs, and $Labels$ values for each model.

3.5 Comparison 2 : Different Type of Cues

When $R = 3$, three cues can be used. However, some of the cues may not be conducive to the prediction but may be a disturbing factor. Therefore, the type of cue used for prediction is important. In this comparison, I suggest two models using different type of cues in order to compare the accuracy. These two models are compared to $M3$, which is the control model for $R = 3$ presented in comparison 1.

3.5.1 Model3-2(M3-2) : Use 1 Cue Which Nearest Cue from the Target

This model has the characteristics of both $M1$ and $M3$. The amount of input is 1 cue, similar to Model 1. However, the robot's sensor range is the same as that of Model 3 ($R = 3$). The model uses input data only about the one observable robot that is closest to the robot which should be predicted. For example, in the Fig. 3.1, if $R = 3$, only $F_3(O_3)$ is used to predict the position of $F_2(P_1)$. The model supposes that the robot most affected by the robot's behavior is the one closest to the robot. The model supposes that the robot most affected by the robot's behavior is the one closest to the robot. According to the assumption, the robot that is most affected by the movement of F_2 is F_3 in the figure. The robot that is nearest to the robot that to be inferred have the most clues about it. The rest of the robots (F_4, F_5) may have fewer clues about the robot that should be inferred. Also, the clues would be delivered in relays, not directly. Therefore, predicting may be possible with only

the information of the robot that is closest to the robot that has to be inferred. To confirm this assumption, I compare the accuracy of Model 3 with the accuracy of this model.

3.5.2 *Model3-3(M3-3) : Use 3 Cues in Same Time Steps*

Model3-3, unlike Model3, uses data at the same time step as input. There are two reasons for using the same time step as input. First, information of different time steps is relatively inaccurate than data of the same time step. To predict $\overrightarrow{P}^{t}_{P_1}$, Model3 uses $\overrightarrow{P}^{t}_{O_3}, \overrightarrow{P}^{t+1}_{O_3}, \overrightarrow{P}^{t+1}_{O_2}, \overrightarrow{P}^{t+2}_{O_2}, \overrightarrow{P}^{t+2}_{O_1}, and\ \overrightarrow{P}^{t+3}_{O_1}$ as *Obs*. However, these values are all relative coordinates of T. Therefore, $\overrightarrow{P}^{t+3}_{O_1}$ is changed to the relative coordinate based on $\overrightarrow{P}^{t+3}_{T}$. However, since T is also continuously moving, $\overrightarrow{P}^{t}_{T}$ and $\overrightarrow{P}^{t+3}_{T}$ are different. $\overrightarrow{P}^{t}_{P_1}$ which has to be predicted is a relative coordinate based on $\overrightarrow{P}^{t}_{T}$, then $\overrightarrow{P}^{t+3}_{O_1}$ based on $\overrightarrow{P}^{t}_{T}$ is required to obtain $\overrightarrow{P}^{t}_{P_1}$.

Therefore, if the data at the different time step is made by the absolute coordinate information, it may be helpful for the prediction, but the data at the different time step may not be helpful for the prediction because it is a relative coordinate based on T at different time. Second, it could more focus on identifying the relationship between the observed robots (O_1, O_2, O_3) using the data of the same time step. If this relationship can be analyzed, O_1, O_2 and O_3 can be seen as a big robot $(\bar{O}_1)$

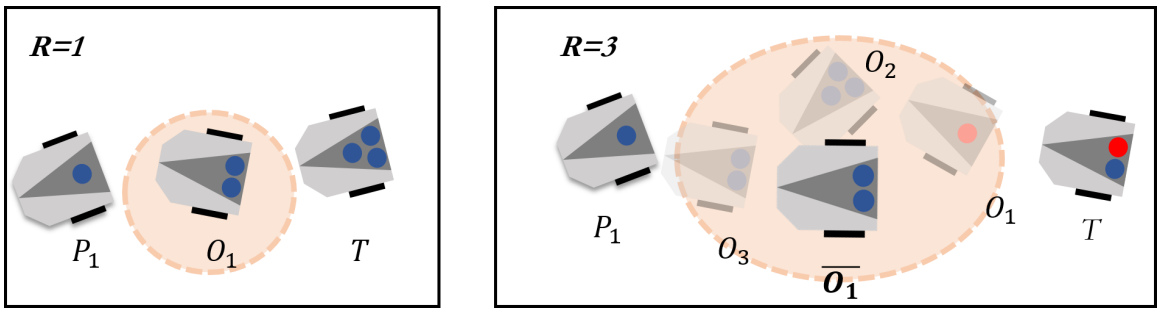


Figure 3.3: Examples of R=1 and R=3

For example, in the Fig. 3.3, when $R = 1$, only information of O_1 is used to predict P_1. When $R = 3$, O_1, O_2, and $O3$ are regarded as one big robot ($\bar{O_1}$) to predict P_1 as in $R = 1$. If three robots can be considered as one robot, it may produce more improved performance. Therefore, the same time step data is input to analyze the relation ship. Table. 3.2 lists the $Hist$, Obs, and $Labels$ values for each model($M3$,$M3-2$,$M3-3$).

	$Hist_{Mx}^{t-r \to t-1} = (X_{Mx}^{t-r}, X_{Mx}^{t-r+1}, \cdots, X_{Mx}^{t-1})$	Obs_{Mx}^{t}	**Labels**
M3	$X_{M3}^{t} = (\overrightarrow{P}_{H}^{t}, \theta_{H}^{t}, \overrightarrow{P}_{F_1}^{t}, \theta_{F_1}^{t}, \overrightarrow{P}_{F_2}^{t}, \theta_{F_2}^{t}, \overrightarrow{P}_{F_3}^{t}, \theta_{F_3}^{t})$	$\overrightarrow{P}_{F_1}^{t}, \overrightarrow{P}_{F_1}^{t+1}, \overrightarrow{P}_{F_2}^{t+1}, \overrightarrow{P}_{F_2}^{t+2}, \overrightarrow{P}_{F_3}^{t+2}, \overrightarrow{P}_{F_3}^{t+3}$	$\tilde{\overrightarrow{P}}_{H}^{t}, \tilde{\theta_{H}^{t}}, \tilde{\theta_{F_1}^{t}}, \tilde{\theta_{F_2}^{t}}, \tilde{\theta_{F_3}^{t}}, \tilde{\theta_{F_3}^{t+1}}.$
M3-2	$X_{M3-2}^{t} = (\overrightarrow{P}_{H}^{t}, \theta_{H}^{t}, \overrightarrow{P}_{F_1}^{t}, \theta_{F_1}^{t})$	$\overrightarrow{P}_{F_1}^{t}, \overrightarrow{P}_{F_1}^{t+1},$	
M3-3	$X_{M3-3}^{t} = (\overrightarrow{P}_{H}^{t}, \theta_{H}^{t}, \overrightarrow{P}_{F_1}^{t}, \theta_{F_1}^{t}, \overrightarrow{P}_{F_2}^{t}, \theta_{F_2}^{t}, \overrightarrow{P}_{F_3}^{t}, \theta_{F_3}^{t})$	$\overrightarrow{P}_{F_1}^{t}, \overrightarrow{P}_{F_1}^{t+1}, \overrightarrow{P}_{F_2}^{t}, \overrightarrow{P}_{F_2}^{t+1}, \overrightarrow{P}_{F_3}^{t}, \overrightarrow{P}_{F_3}^{t+1}$	

Table 3.2: Hist,Obs,Labels According to the Model(M3,M3-2,M3-3)

3.6 Comparison 3 : Different History Length

The control model($M1$) uses global information as *historical* data. $Hist$ is the sequential data that have the position and orientation of the robot from time $t - r$ to $t - 1$. These data pass through the LSTM layer and consist of one layer. The effect of the sequences may be different according to the history length and it may affect the prediction result. To compare the results according to the history length, three models with different history lengths were designed based on the structure of Model 1. The control model has a history length of 10, that is, data of 10 time steps is entered as history. In addition, two models with history lengths of 6, and 4 are

created. I compare the results of these three models to analyze the effect of history length on the results. The *historical* data and *Obs* data of the model Hx are as follows. x is history length.

$$Hist_{Hx}^{t-x\to t-1} = (X_{Mx}^{t-x}, X_{Mx}^{t-x+1}, \cdots, X_{Mx}^{t-1}) \quad , \quad X_{hist}^{t} = (\overrightarrow{P}_{H}^{t}, \theta_{H}^{t}, \overrightarrow{P}_{F_1}^{t}, \theta_{F_1}^{t})$$

$$Obs_{Hx}^{t} = (\overrightarrow{P}_{F_1}^{t}, \overrightarrow{P}_{F_1}^{t+1})$$

3.7 Recursive Prediction Approach

Follower robots move using the same motion rules. Thus, the above models, which analyze this rule and make predictions, can apply it to all robots. In this experiment, an approach is used to infer information about the entire robot using the prediction model recursively. For example, in Fig. 3.1, suppose that $R = 1$ and the $M1$ is used, then T infers $\overrightarrow{P}_{F_4}$ using the $\overrightarrow{P}_{F_5}$. The predicted $\overrightarrow{P}_{F_4}$ and $\overrightarrow{P}_{F-5}$ can be used to predict $\overrightarrow{P}_{F_3}$. In addition, $\overrightarrow{P}_{F_2}$ can be predicted by using $\overrightarrow{P}_{F_3}$ and $\overrightarrow{P}_{F_4}$. The prediction result can be reused to predict repeatedly up to $\overrightarrow{P}_{H}$.The use of the predicted result for the next prediction is called relay prediction. In the above example, relay prediction is done twice to obtain $\overrightarrow{P}_{F_3}$, so relay is 2 ($Relay = 2$).

In summary, it is possible to infer global information about the entire robot by re-prediction based on the predicted results. Both the above comparison models and the control model are applied to the relay prediction approach to predict the position up to H.

Chapter 4

EXPERIMENT DESIGN1 : REAL ROBOTIC PLATFORM

In order to implement the swarm robots that make chain formation using real robots, three equipment should be dynamically operated. A commercial robot called *Thymio* is responsible for the role of a real robot in the experiment. *Thymio* is moving by adjusting the speed of two wheels to reach the desired target. In the swarm robotics, global communication and central computer does not exist. However, the sensor of the robot has a low performance to acquire local information. Therefore, adding two equipment to the experiment to get exact local information and deliver it to the robot. A central computer uses a camera to capture the position and the orientation of the robot and changes the information to the relative coordinates and sends it to the robot. *Raspberry pi* plays the role of sending information from the central computer to the robot.

4.1 Platform Component 1 : Physical Robot (Thymio & Raspberry Pi)

4.1.1 Role of Thymio

Thymio acts as a real robot. *Thymio* is a two-wheel robot which determines its direction and speed using the speed of the left and the right wheel. These wheels can rotate independent of each other, so if they will rotate in different speed the robot can go either straight or turn. Fig. 4.1 shows about the notation of each part of the robot and how to express the position and the orientation on the coordinate plane. (x, y, θ) indicates the current position and orientation. $(\dot{x}, \dot{y}, \dot{\theta})$ is the desired value that the robot wants to move. The robot needs to find v_r and v_l to move to $(\dot{x}, \dot{y}, \dot{\theta})$. the velocity$(v)$ and the angular velocity(w) can be found using (x, y, θ) and $(\dot{x}, \dot{y}, \dot{\theta})$.

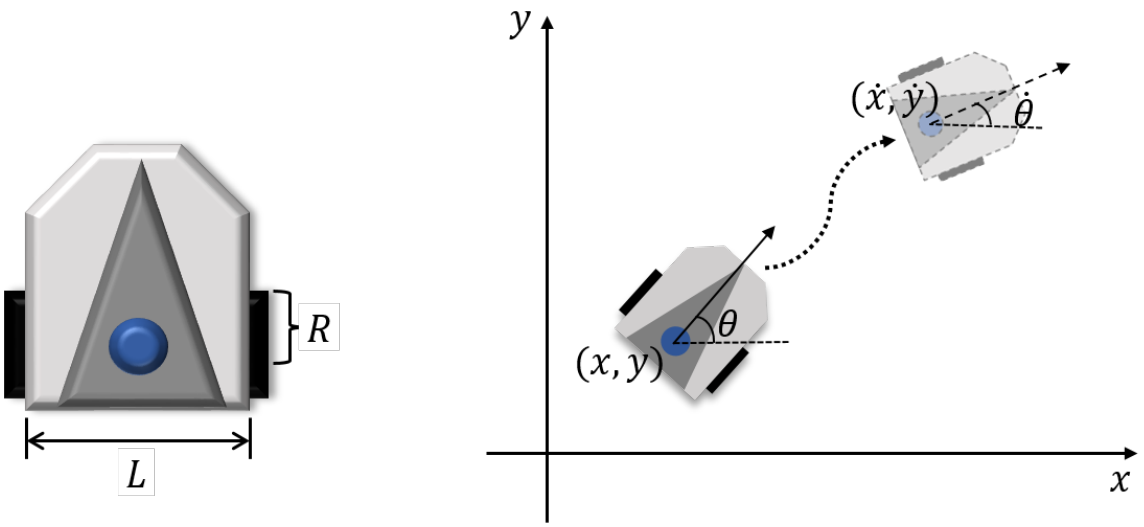


Figure 4.1: Notation of Each Part of the Robot and How to Express the Position and the Orientation on the Coordinate Plane.

Therefore, when v and w are given, it should be able to find v_r and v_l. To obtain this, a *Kinematic equation* for differential drive robot (2) is used. The *Kinematic equation* for differential drive robot can be derived from *Unicycle kinematic* (1).

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = w \end{cases} \quad (1) \quad \Rightarrow \quad \begin{cases} \dot{x} = \frac{R}{2}(v_r + v_l)\cos\theta \\ \dot{y} = \frac{R}{2}(v_r + v_l)\sin\theta \\ \dot{\theta} = \frac{R}{L}(v_r - v_l) \end{cases} \quad (2)$$

Using the equation (1) and (2), v and w can be expressed using v_l and v_r as shown in the equation (3). Using this, the equation (4) can be obtained. The parameters R and L are known constant values. Therefore, using equation (4), v_r and v_l can be calculated when v and w are given. *Thymio* uses the equation (4) to calculate v_r and v_l.

$$\begin{cases} v = \frac{R}{2}(v_r + v_l) \rightarrow \frac{2v}{R} = v_r + v_l \\ w = \frac{R}{L}(v_r - v_l) \rightarrow \frac{wL}{R} = v_r - v_l \end{cases} \quad (3) \quad \Rightarrow \quad \begin{cases} v_r = \frac{2v+wL}{2R} \\ v_l = \frac{2v-wL}{2R} \end{cases} \quad (4)$$

4.1.2 Role of Raspberry Pi

As mentioned above, there are some difficulties in experimenting because *Thymio* is a robot made for educational purposes. First, in order to operate the robot, only the programming language provided by *thymio* called *Aseba* must be used. Second, to use multiple *thymio* wirelessly, a computer is required as many times as the number of *thymio*. To solve these two problems, *Raspberry pi* is used in the experiment. Fig. 4.2 is a physical connection between the *Raspberry pi* and *thymio*.

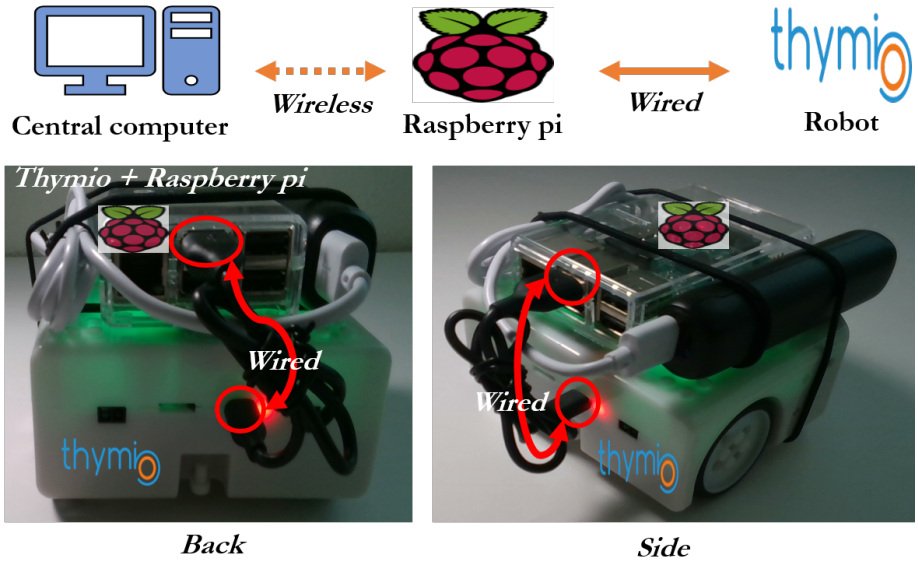


Figure 4.2: Physical Connection Between Raspberry Pi and Thymio

Raspberry pi has two roles. First, *Raspberry pi* acts as a bridge between *Thymio* and central computer. *Thymio* and *Raspberry pi* are wired and *Raspberry pi* and a central computer are wirelessly connected. Once *Thymio* and *Raspberry pi* are physically connected, the *Raspberry pi* can control *Thymio* using the kernel. In addition, *Raspberry pi* and the central computer can exchange information between the two via socket communication using *the socket python library*. Fig. 4.3 describes the procedure of socket communication. If the *Raspberry pi* and the center computer

are in the same network, socket communication can be used. The server waits until the client requests access, then sends and receives data after they are connected. In the experiment, the central computer acts as a server, the *Raspberry pi* acts as a client, and the local information provided by the central computer is provided through socket communication. This information is passed to *Thymio* after further calculation.

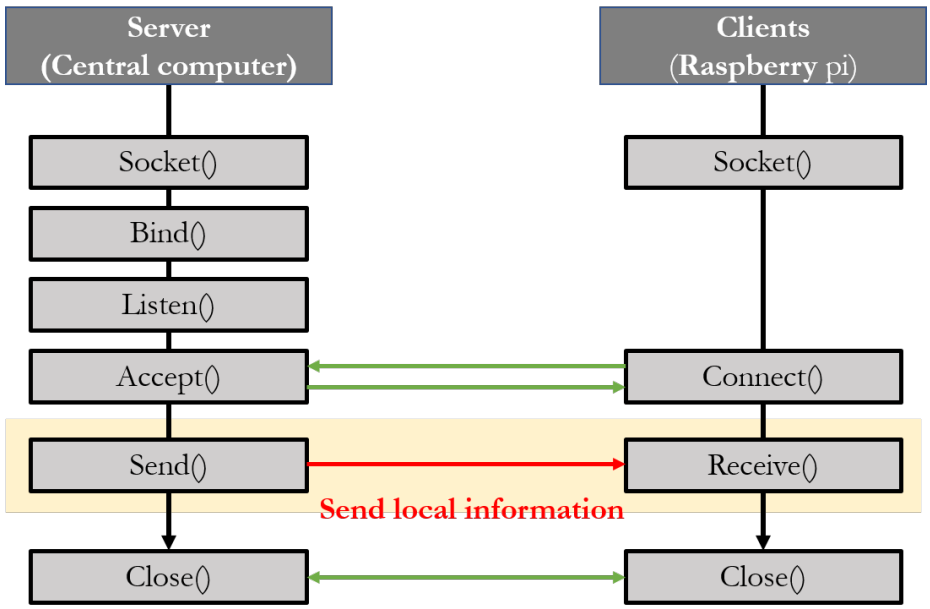


Figure 4.3: Socket Communication Between the Server(Central Computer) and the Clients(Raspberry Pi & Thymio)

Second, the *Raspberry pi* acts as the brain of the robot. The *Raspberry pi* calculates the next target position and orientation to be reached using information from the central computer and the motion rule for getting robots to move. Using the calculated next target position and the current position, the *Raspberry pi* obtains the v and w to reach the target and passes these values to the *Thymio*.

4.1.3 Motion Rule

Based on local information, the robot sets the next target point according to the motion rule and moves to it. Three different motion rules are used. The motion rule (1) is used by the head robot(H) at the front of the line, the motion rule (3) is used by the tail robot(T) at the back of the line, and the motion rule (2) is used by the follower robots(F) located in the middle of the line. All motion rules are suggested in Choi *et al.* (2017).

$$\dot{\vec{P_H}} = \vec{P}_{Target} - \vec{P_H} \quad (1)$$

$\vec{P_R}$ is (x, y) indicating the position of the robot(R) on the coordinate plane. $\vec{P}_{Target}$ is the (x, y) of the target that the head(H) should move. In accordance with the rule (1), the H always sets the following position ($\dot{\vec{P_H}}$) so that it is close to the target.

$$\dot{\vec{P}}_{F_i} = k_f((||\vec{P}_{F_i} - \vec{P}_{F_{i-1}}|| - d)(\vec{P}_{F_{i-1}} - \vec{P}_{F_i}) + (||\vec{P}_{F_i} - \vec{P}_{F_{i+1}}|| - d)(\vec{P}_{F_{i+1}} - \vec{P}_{F_i}) \quad (2)$$

The followers(F_x) use the rule (2) to calculate the position to be moved next. k_f is a scalar value that determines the weight to be reflected in the next action among the front and rear robot. d indicates the distance that robots should maintain. Since $||\vec{P}_{F_i} - \vec{P}_{F_{i-1}}||$ is the current distance between i and $i-1$ robots, and d is the ideal distance that the robots should maintain, the closer the distance between the two robots is to d, the closer to $||\vec{P}_{F_i} - \vec{P}_{F_{i-1}}|| - d$ is 0. $(\vec{P}_{F_{i-1}} - \vec{P}_{F_i})$ is a vector that induces the i robot to calculate the next position closer to the $i-1$ robot. Therefore, the closer the distance between robots is to the ideal distance (d), the smaller $(||\vec{P}_{F_i} - \vec{P}_{F_{i-1}}|| - d)(\vec{P}_{F_{i-1}} - \vec{P}_{F_i})$ value, and conversely, the farther away from the ideal distance(d), the larger $(||\vec{P}_{F_i} - \vec{P}_{F_{i-1}}|| - d)(\vec{P}_{F_{i-1}} - \vec{P}_{F_i})$ value. Likewise, $(||\vec{P}_{F_i} - \vec{P}_{F_{i+1}}|| - d)(\vec{P}_{F_{i+1}} - \vec{P}_{F_i})$ allows the i robot and $i+1$ robots to maintain an ideal distance(d). Therefore, the i robot moves to maintain the ideal distance from the front and rear robots ($i-1$ and

$i+1$ robots).

$$\dot{\vec{P}_T} = k_f((||\vec{P}_T - \vec{P}_{F_1}|| - d)(\vec{P}_{F_1} - \vec{P}_T)) \quad (3)$$

Since the tail robot(T) have no robots behind, tail robot(T) only needs to consider the robot in front. Thus, the tail(T) follows the rule (3), which removes the part about the rear robot from the rule in (2).

4.2 Platform Component 2 : Central Computer and Camera

Generally, there is no global coordinating system in the swarm robot. Swarm robots use only local information such as position information of adjacent robots detected within sensor range. Therefore, it is important to accurately capture local information in swarm robots. The sensors in the *Thymio* are not performing well. The sensor radius is very small less than 5cm, and it is also difficult to estimate a precise position even if detected by the sensor.

To compensate for these shortcomings, the central computer and camera are used in the experiment. However, the central computer does not deliver global information to the robots and also it does not operate them. The central computer calculates the current position and orientation of the robots using images captured by the camera. Information of the robots within the sensor range of each robot is changed to a relative coordinate and transmitted to the robot. For example, if there are robots as shown in the Fig.4.4 and the robot's sensor range is equal to the orange circle, F_2 robot

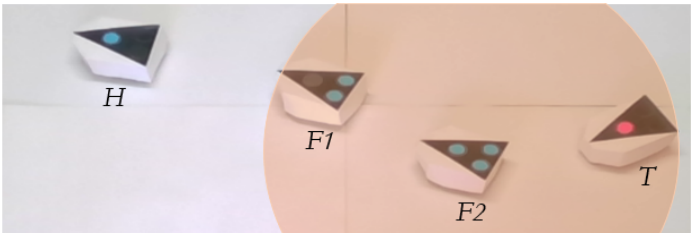


Figure 4.4: An Example of Observable Robots According to Sensor Range

receives information about robot F_1 and T. The position coordinates of F_1 and T are changed to relative coordinates relative to the F_2 robot, and the orientations are the absolute coordinate values relative to the $x - axis$. The central computer delivers these position and orientation only to F_2

4.2.1 Role of the Central Computer

The central computer replaces the sensors in the *Thymio* to find the position and direction of adjacent robots. The central computer converts the images received from the camera into an (x, y) coordinate system and finds the coordinate values corresponding to each robot. Also, the orientation that each robot is looking at is determined by using the $x - axis$ of how many degrees it is tilted. The coordinates and orientation values of adjacent robots, which are assumed to be observable by the sensors, are changed to the relative coordinates based on each robot. The reason for changing to a relative coordinate is because it is assumed that the global coordinate does not exist in the swarm robot. Therefore, to implement a more realistic swarm root, relative coordinate values, not absolute coordinates, are required (Tan and Zheng, 2013). Then, through socket communication with the *Raspberry pi*, the corresponding relative position is delivered to the *Raspberry pi* so that the *Raspberry pi* and the *Thymio* can calculate the target value that should be reached.

4.2.2 Methodology for Identifying Robots in Video

The central computer takes a series of images refinement process in real time. It distinguishes the robot and finds the location and direction of the robots. This process requires the technologies of computer vision. Vision technology is implemented using *open-cv*, one of the *Python* libraries. As shown in the Fig. 4.5, the robot's position and direction are obtained in the image after four step processes.

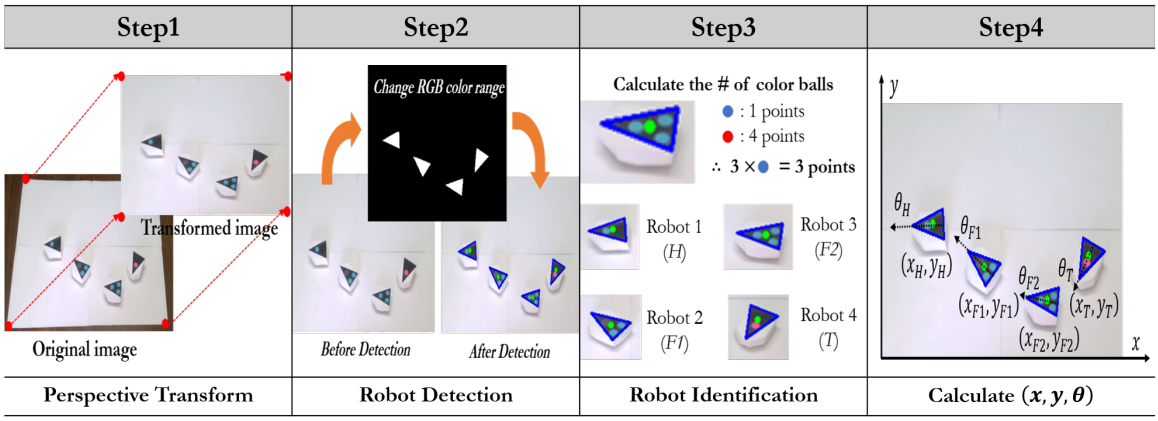


Figure 4.5: The Process of Detecting a Robot in a Video Frame

Step 1. Perspective transformation : Perspective transformation is one of the image warping technologies, which changes the pixel located in (x, y) to $(\acute{x}, \acute{y})$ through geometric deformation. Unless the Camera is perpendicular to the Arena, the Arena is filmed tilted. As shown in the left figure in Fig. 4.6, the arena is tilted in the shape of a polygon, not a rectangle. Therefore, it is necessary to restore this

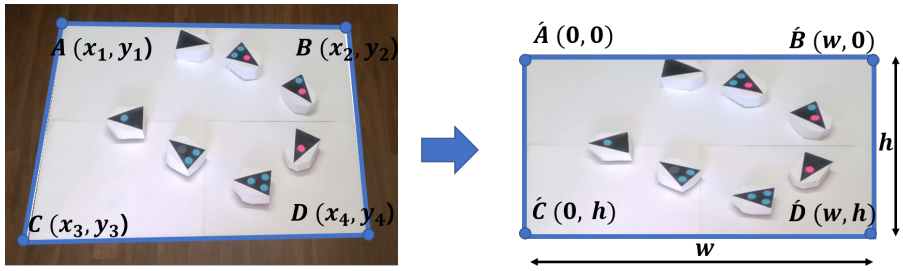


Figure 4.6: An Example of Perspective Transformation

tilted image as if it is taken vertically. Four points corresponding to the edges of the arena are selected as A, B, C, and D in the Fig. 4.6 and $\acute{A}, \acute{B}, \acute{C}$, and $\acute{D}$ are set to fit the size of the arena to perform warping. By doing so, the tilted image is changed to the similar image as the one taken from the vertical. Therefore, it has become possible to determine the position of the robot using the pixel.

Step 2. Robot detection : Finding the robots in images that have undergone

perspective transformation is another challenge. Among the many ways to find objects in the images, this experiment selects a color-based classification method. The robots are covered with the case. The left image in the Fig. 4.7 is for comparison between a covered robot and a uncovered robot. When the camera takes the covered robot, it is like the right image in the Fig. 4.7. If all other colors are removed from

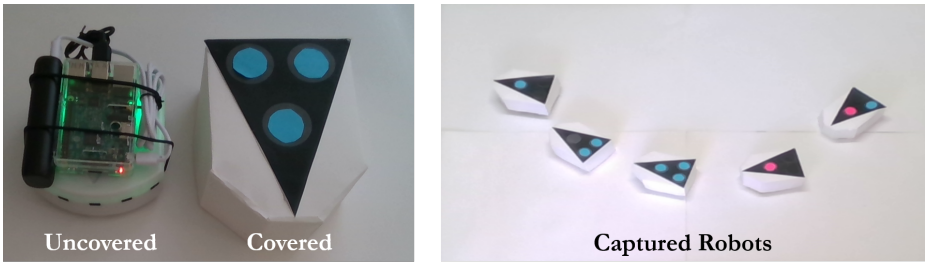


Figure 4.7: The Appearance of the Robot Depending on Whether It Is Covered & A Picture of Covered Robots

the image except the black part so that only the black part is detected in this image, only the black triangle part is left as shown in the Fig. 4.8. If the area of each triangle exceeds a certain size, it is set to be recognized as a robot. Therefore, a total 4 robots are detected in the Fig. 4.8.

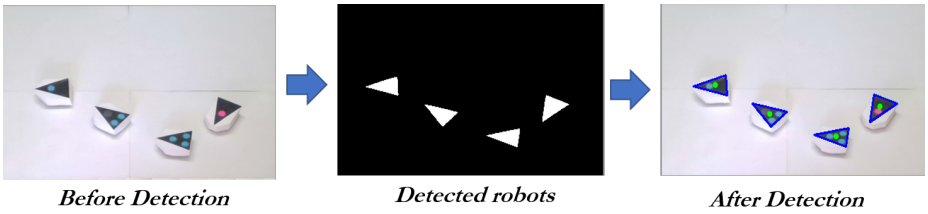


Figure 4.8: The Process of Detecting a Robot by Adjusting the Color Range

Step 3. Robot identification : In the experiment, identification of the robots is determined by the color and the number of circles in the triangle. The blue circle represents one point and the red circle represents four points. For example, the ID of F_1 robot with two blue circles inside the triangle is 2. Total seven robots are used

in the experiment, and each robot's ID can be identified as shown in the Fig. 4.9. Using this method, the ID can be effectively expressed even as the number of robots increases.

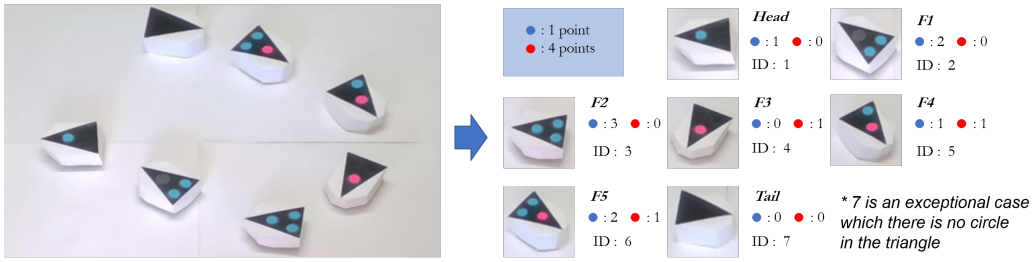


Figure 4.9: An Example of Robot Identification

Step 4. Calculate (x, y, θ) : Once the detection and identification of the robot has been completed, the robot's position (x, y) and the orientation (θ) should be calculated. In the case of the position, the number of horizontal and vertical pixels of the image is set proportionally to the size of Arena in Step 1. The number of pixels is set in units of 1 cm per 3 pixels. Therefore, the pixel value at which the robot is located can be considered as the position of the robot.

The exact position of the robot is determined by the center of gravity of the triangle. The hard cover of the robot is made so that the center of the robot is aligned with the center of gravity of the triangle. Therefore, in the process of detecting the triangle in Step 2, the center of gravity is obtained, and the (x, y) value of the center of gravity will be the current position coordinate of the robot. The vertex and center of gravity of the triangle are used to find the orientation. Draw a line connecting the vertex at the front of the robot to the center of gravity at the front of the robot. The angle of intersection between the line and the x-*axis* is the orientation the robot is pointing. Through this method, (x, y, θ) of the robot can be obtained.

Fig. 4.10 is the flow chart of the platform. *Thymio*, *Raspberry pi*, and the central computer operate according to this flow.

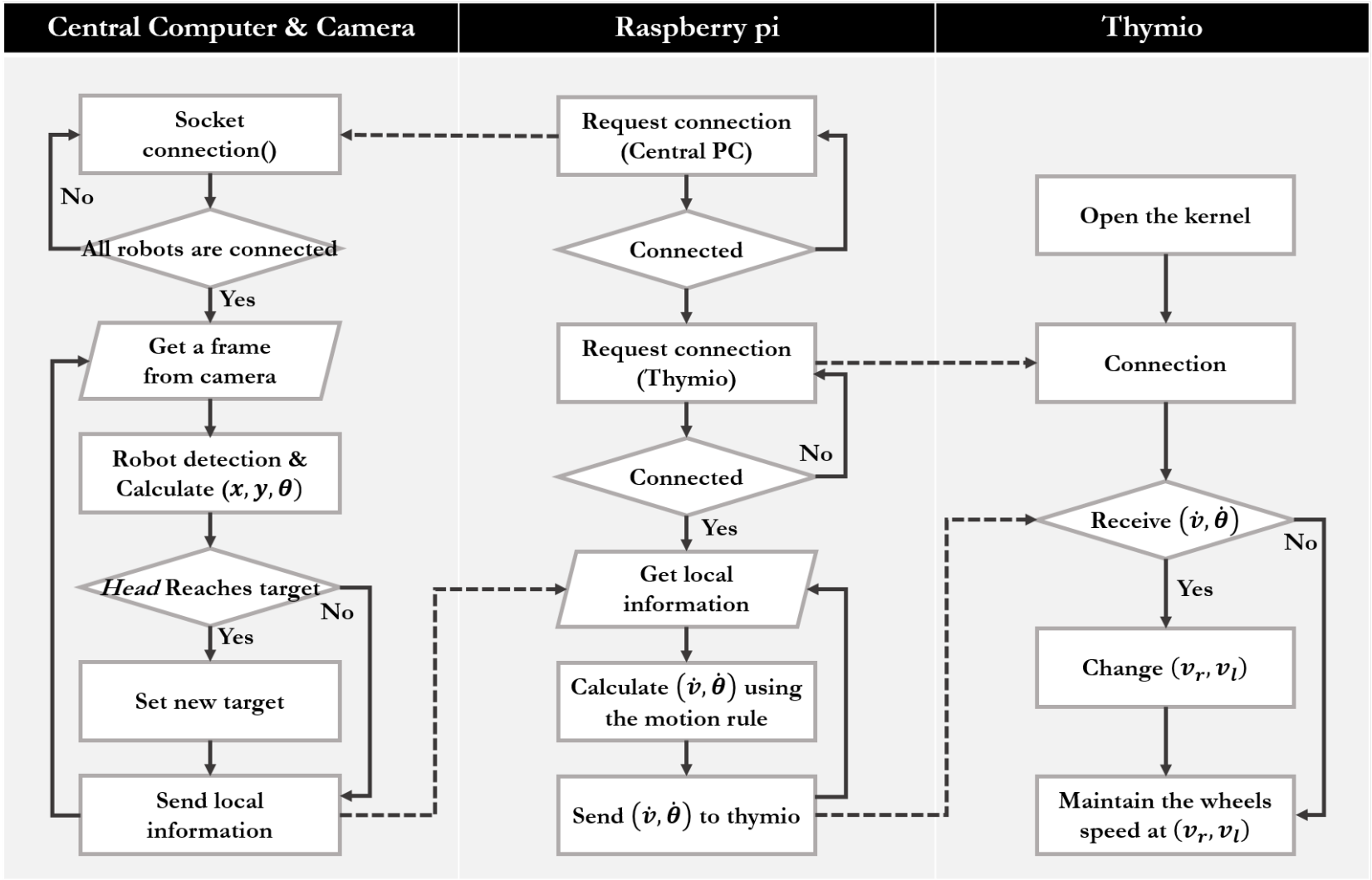


Figure 4.10: Platform Operation Flow Chart

Chapter 5

EXPERIMENT DESIGN2 : DATA COLLECTION

The data, which used to predict the position and orientation of the remote robot, is the position and orientation of the adjacent robot ($Hist$ and Obs). To collect the data, the robots move in a row on the arena, and the cameras records them. Based on the video filmed, the position of each robot and the orientation are analyzed. Therefore, at the arena, a head robot (H) needs to move randomly, without bumping into other robots. In consideration of this, an algorithm(ψ) to set the target point is applied. The algorithm(ψ) is designed to set random target points by considering the robot's progress direction and location of adjacent robots. In addition, the data is collected in case of using three, four, five and seven robots each. The data about three, four, and five robot cases is used for training and validation, while the data about seven robot case is used for testing.

5.1 Physical Platform

Arena, an area where robots operate, is a flat floor with a specification of $2.8m \times 2.2m$. The camera captures the whole arena, and the central computer only cuts the arena region from the filmed image and changes the image size 840×660 $pixels$(1cm per 3pixels). Therefore, the location of robots can be obtained with (x, y) coordinate using pixels. The camera shoots at 12 frames per second (12 fps). The robot receives the position of the adjacent robot once every 0.25 seconds and changes the speed of wheels(v_r and v_l) to move to a new target position derived by the motion rule. Since predictions are to be made at 0.5 second intervals, the robots' position and orientation are collected and stored at the same intervals.

5.1.1 *Problems When Setting Target Coordinates Arbitrarily*

The head robot(H) is located at the head of the team and serves to move the team members to a designated destination. The head robot(H) applies a different motion rule than the followers(F_x), which moves the robot to the target point. To collect data continuously, when the head(H) reaches the target point, a new target point is set so that the head robot and other robots could continue to roam around the area. The data used in experiments should be objective and should not be manipulated or changed due to external intervention. In the process of collecting data, if a person's subjective or behavior pattern is included in the data collected, the data becomes changing and cannot create accurate results.

Thus, when the central computer set the target point which the head robot aims, the target point must be set in an environment where there is no external intervention as possible. The way to set the target point without external intervention is to always set the target point randomly. However, this method creates two problems. First, collisions with other robots often occur while moving to a set random target. The motion rule used by H does not consider the position of adjacent robots, but only considering getting close to the target. As a result, the robots have often been placed on the path of moving to the target point. The picture on the left in Fig. 5.1 is an example. In the figure, the yellow star is the next target point. H moves along the

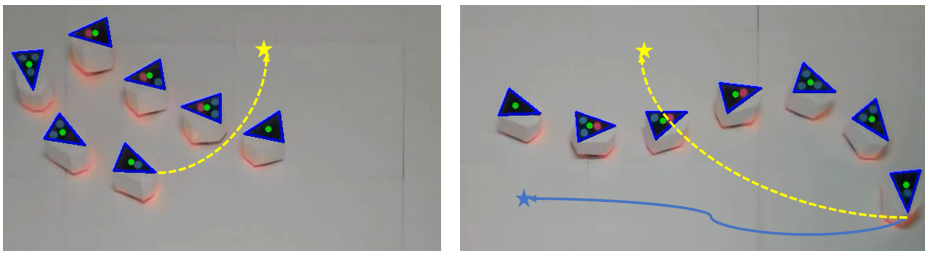


Figure 5.1: Examples in Which the Target Is Set Incorrectly and Make Crashes.

shortest path (dotted line) to the target point. However, other robots are placed on the path along which H moves and they will be collided. Second, a next target point is set to a point that is difficult to physically move from the current location. When the head(H) is in the corner or boundary of the arena, a collision between robots or moving out of the arena occurs while moving to the next target point. In the right picture of Fig. 5.1, robots are located in the corner. If H moves along the blue solid line, collision between robots can be avoided. However, the target point (the yellow star) is selected differently from the solid line direction, and as a result, H moves to the dot line and the robots collide with each other. The reason why the collision occurs frequently is that the position of the other robots is not considered. To solve the problems and choose targets randomly, the following algorithm is applied.

5.1.2 Target Point Setting Algorithm

This algorithm is the same algorithm used by Choi *et al.* (2020). This algorithm is conducted by the following steps.

Step 1. Divide the arena into small areas of n by m. Arena consists of $m \times n$ small cells $c = \{c_{ij} | 1 \leq i \leq m, 1 \leq j \leq n\}$. In the experiment, $m = 6$ and $n = 4$, the size of one cell is $43 \times 49cm$

Step 2. Considering the movement of the robot, select candidate cells to set the next target point. The movement of the robot is determined by using the H's move from the previous location cell($M[0]$) to the current location cell($M[1]$). $M[1]$ represents the cell number where H is currently located ($M[1] = \{c_{ij} | i = x_{now}, j = y_{now}\}$, $M[0] = \{c_{ij} | i = x_{prev}, j = y_{prev}\}$). For example, if H is currently located at C_{12}, then $M[1] = C_{12}$. Candidate cells are selected by considering $M[0]$ and $M[1]$. First, find the cell (M[1]) where the H is currently located. The value in $M[1]$ is shifted to $M[0]$.

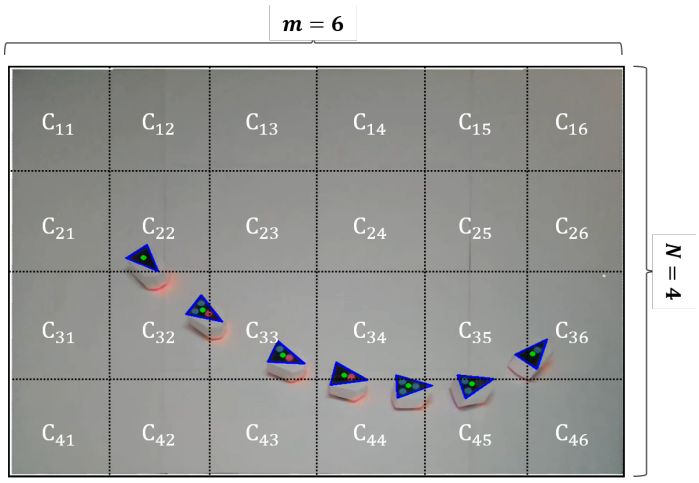


Figure 5.2: An Example of Expressing Arena as Cells

Cells adjacent to $M[1]$ that are not adjacent to $M[0]$ are considered as candidates.

$$C_{candidates} = \{C_{ij} | Near(M[1], C_{ij}) \cap\ !Near(M[0], C_{ij})\}$$

$$Near(C_{ij}, C_{mn}) = \begin{cases} True & if\ |i-m| \leq 1\ and\ |j-n| \leq 1 \\ False & else \end{cases}$$

Considering $M[0]$ and $M[1]$, there are two possible cases. The first is moving straight (The left figure in Fig. 5.3) and the second is moving diagonally (The right figure in Fig. 5.3). However, if the robot moves to the outer boundary area, the candidate cells may be set as an area that the robots cannot move. For example, as in the Fig. 5.4, H moves to the vertex of arena or to the edge. In this case, the candidate cell does not exist when determining the candidate cell as described above. In this exceptional case, movable cells are selected as candidate cells among cells adjacent to $M[1]$ without considering $M[0]$. The candidate cells are selected as shown in the Fig. 5.4. The reason for selecting candidates in this way is as follow. H is more is more likely to hit an adjacent robot when it rotates sharply more than 90° from its current direction. Since the robots are located behind H, when the robot spins sharply by more than 90°, the robot will return to the path it came from, increasing the

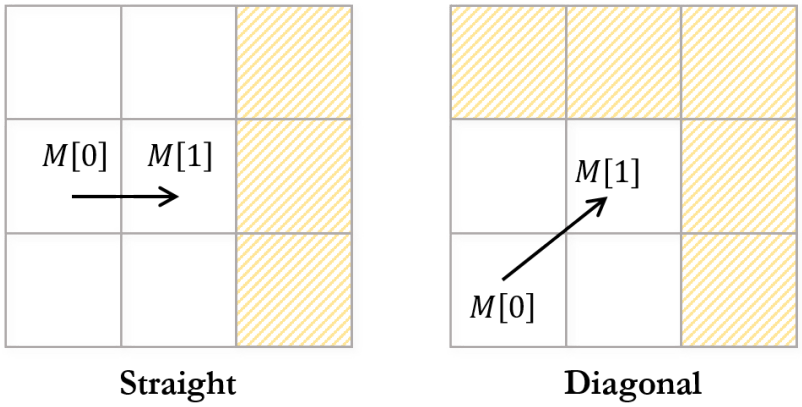


Figure 5.3: Cases that can be set as candidates cells considering the movement

probability of collision. Therefore, the above rule is used to set the target considering the current movement direction.

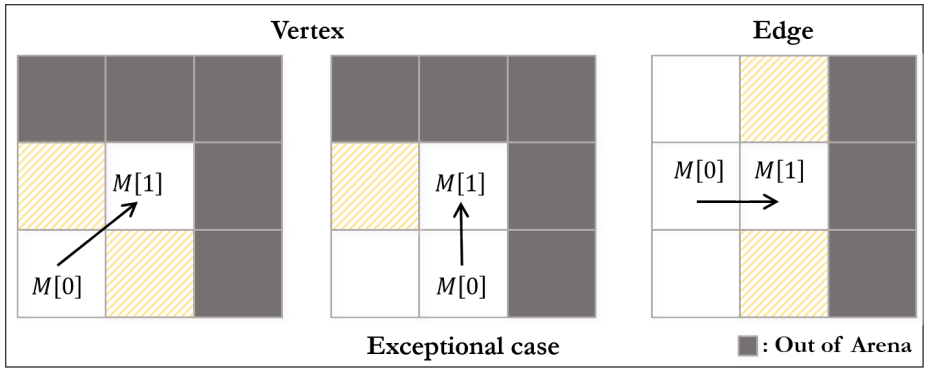


Figure 5.4: Cases When Special Candidate Cells Have to Be Designated Due to Unmovable Areas

Step 3. If another robot is within a candidate area, the area is excluded from the candidate list. In the left image of the Fig. 5.5, the three hatched cells represent the candidate cells obtained in Step 2. When the next target is selected in the cell where the other robot is located, the collision probability increases, so the cell (the red surrounded cell in the left figure in the Fig. 5.5) is excluded from the candidate list. Thus, the two hatched cells become the candidate in the right figure in the Fig. 5.5.

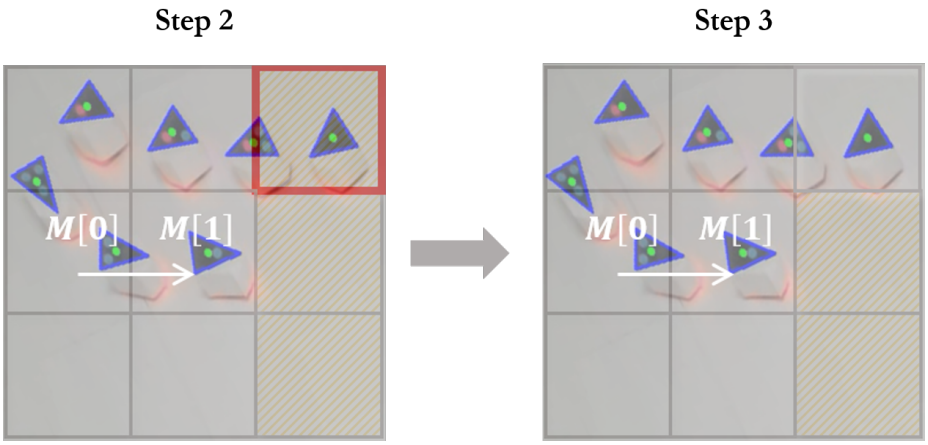


Figure 5.5: An Example of a Candidate Cell Being Excluded from a Candidate Due to Other Robots

Step 4. Choose one of the candidate cells selected through the process up to Step 3. The probability of choosing are all the same. ($P_{c_{ij}} = \frac{1}{n}$, $n = \#$ of candidate cells, $c_{ij} \in c_{candidates}$). Finally, the coordinates (x, y) are randomly selected in the chosen cell.

Step 5. When H arrives at the target, repeat the Steps 2, 3, and 4.

5.2 Collected Data for Training, Validation, and Testing

The total amount of data collected is shown in the Table. 5.1. The data is recombined into *Sample* and *Instance* units for purposes. The data about 3-robot group, 4-robot group, and 5-robot group are used to train and validate neural network models. The data about 7-robot group is used to test the models.

5.2.1 Sample, Instance

Sample is the unit used in the training and it is the minimum input unit for the neural network models. *Sample* is classified for predicting one time. In other words, each sample make one output(label). For example, Data at $t-10$ to $t+1$ time steps

	Duration(minutes)	# of samples	# of instances
3 robots	131.0	15,178	-
4 robots	133.7	15,504	-
5 robots	139.6	15,784	-
7 robots	84.9	-	247

Table 5.1: Description of Data Collected from 3,4,5,and 7 Robot Teams

are required to predict the position of P_1 at time t using $M1$. *Sample* is composed by grouping the data needed for one prediction. *Sample* is composed as the Fig. 5.6.

In the figure, the yellow hatched box is the data used for $Hist$, the blue hatched box is the data used for *Obs*, and the red hatched box is the labels. In each table, columns represent time steps, and index represents the robot's position or orientation.*Sample* means the total value corresponding to the table.

$$Sample\ i : \left\{ \overrightarrow{P}_R^{t-l \to t+i}, \theta_R^{t-l \to t+i} | R \in \{P_1, O_k | 1 \leq k \leq i\}, l = history\ length \right\}$$

Since the information used for prediction differs according to models, the sample is configured differently. Samples used by $M1$, $M2$, and $M3$ use data of data1, data2, and data3, respectively. $M3-2$ and $M3-3$ use the information contained in the sample of $M3$, then they use the same sample used by $M3$ to training. When comparing models with different history lengths($H10$,$H6$,$H2$) use the same sample used in $M1$. They use the adjusted data by the length of history in the same sample and use it for training. For example, $H10$, $H6$, $H2$ use the same samples, but $H10$ uses $0-9$ time steps data as $Hist$ to predict $\overrightarrow{P}_{P1}^{10}$, H6 uses $4-9$ time steps data to make a prediction.

$Instance$ is the unit used in the test. $Instance$ is used to make 15 consecutive predictions. The models used in the experiment use a history of 10 steps and predict the position and orientation for the next 15 steps. Therefore, 26 time steps of data

are configured into one instance. In addition, an interval of 7 seconds (14 time steps) between the instances is placed to ensure the independence of the instance by eliminating the continuity between them.

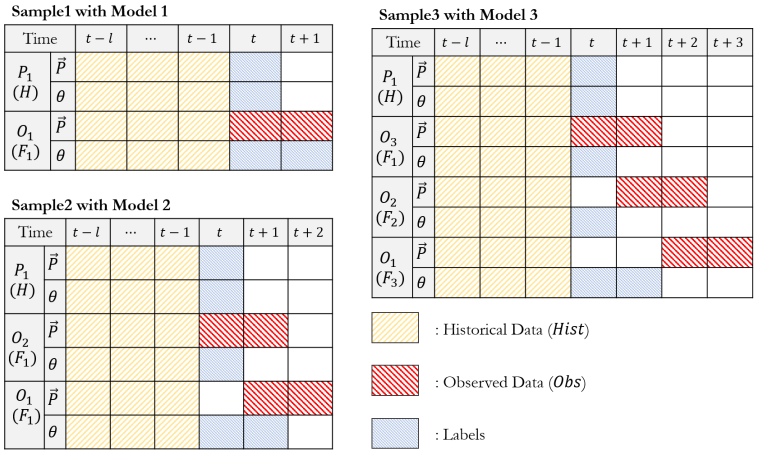


Figure 5.6: Sample Composition According to the Prediction Model

5.2.2 Training, Validation, Test

For training and test the models, the data is divided into train, validation, and test. The data used for training and validation differs depending on the model. $M1$, $M2$, and $M3$ proposed for Comparison 1 (Comparison according to the difference number of cues) use each of $Sample1$, $Sample2$, and $Sample3$. $M3$, $M3-2$, and $M3-3$, which are proposed for Comparison 2 (Comparison according to the difference type of cues), use $Sample3$. The proposed models $H10$, $H6$, and $H2$ for Comparison 3 (Comparison according to history length) use $Sample1$. 80% of sample is used for training and 20% is used for validation. Validation is only used for testing during training to prevent overfit of the model. The models save the weights when the validation error is the lowest. To make comparisons under equivalent conditions, the same data is used for the test. The test uses instances created with data made using 7 robots. The total number of instances used for the test is 247.

Chapter 6

RESULTS

The first goal of the experiment is to implement team-robot using real robots based on the simulation results of Choi *et al.* (2017). Using the above methods and equipment, team-robot is successfully implemented. *Thymio* and *Raspberry pi* act as robots and the central camera and central computer act as sensors for the robots. Although a central computer is used, the central computer's role is only to transmit information about adjacent robots. Thus, the robots have the characteristics of SR, then they are decentralized and relatively free to increase or decrease the number of robots in team. At first, three robots are used to move in chain formation, and then expand to seven, forming a longer chain.

The second goal of the experiment is to compare the accuracy of predictions based on differences in the information used when making predictions about remote robots using information from adjacent robots. The comparison is conducted in three ways. The first is the comparison of the results when the number of cues used is different. The second is comparing to the differences in results depending on the type of cues used. The third is to compare the difference the results when the model's history length is different. The deep neural network models are used to predict the position and direction of the remote robot using the data of the robots moving in the arena. Each model is separately trained 7 times and the performance of the model is considered as the average value of 5 cases excluding the two cases with the best performance and the worst performance.

6.1 Comparison 1

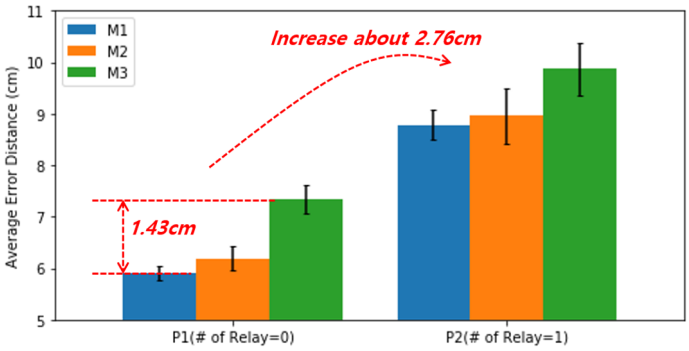
The amount of information available may vary according to the observation range (R) of the robot. The information of observable robots is called as cue, and the accuracy of prediction is compared when the number of cues is 1, 2, and 3 each, that is, when the robot can observe 1, 2, or 3 adjacent robots each. Model1 (M1), Model2 (M2) and Model3 (M3) use 1, 2 and 3 cues respectively. There are two ways to compare the results. First, the prediction of the robot closest to T among the robots outside the observation range (R). This is a prediction without recursive (relay) prediction (prediction of a robot farther away using the predicted result). In other words, the prediction result is the first prediction that is not input again as information for prediction. Thus, this comparison can compare the baseline performance of the model.

Second, it is a comparison of accuracy according to the robot. Without considering the relay, the target classifies the robot according to the distance from T (H, F_1, F_2, etc.), and compares the results for each model according to the target. This comparison is effective to compare the accuracy of the predictions for the entire robot.

6.1.1 Comparison According to the Number of Relays

In order to apply the recursive prediction approach mentioned in the Method chapter, relay prediction which uses a prediction result for the next prediction is used. In order to predict P_2, P_1 must be predicted first and the predicted value of $P_1(\widetilde{\overrightarrow{P}}_{P_1}, \tilde{\theta_{P_1}})$ must be used for P_2 prediction. The use of this predicted result in the next prediction is referred to as relay. (relay = 1 to predict P_2)

The comparison of the results according to the number of relays for each prediction



p-value / Robot	M1 vs M2	M1 vs M3	M2 vs M3
P1	0.0134	0.0000	0.0000
P2	0.4628	0.0003	0.0061

Figure 6.1: Average Prediction Error for Each Model According to the Number of Relays and $p-values$ Between Models

model is as follows. P_1 means robot with $relay = 0$. The robots corresponding to P_1 are different for each model. As shown in Fig. 3.1, P_1 is F_4 in $M1$, P_1 is F_3 in $M2$, and P_1 is F_2 in M_3. Fig. 6.1 is a graph of prediction error for each model according to the robot and a table that summarizes the $p-values$ between models. The graph shows the mean of the prediction error for the robots by models.

In the cases of P_1 and P_2, the average distance error is $M1$ the smallest and $M3$ the largest. $(M1 < M2 < M3)$ $M1$ performed 10% to 21% better than $M3$. In all cases, the $p-value$ between $M1$ and $M3$ is lower than $0.05/3$. It can be said that the performance of $M1$ is better than that of $M3$ with a reliability of 95% or more.

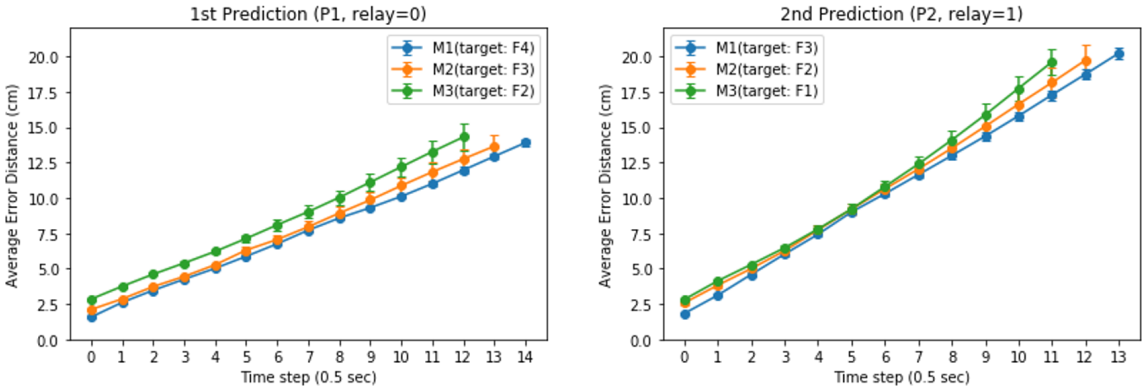


Figure 6.2: Detailed Comparison of Prediction Results for P_1 and P_2 According to Time Step

The graphs in the Fig. 6.2 compare the predicted error values for P_1 and P_2 each in detail with time-step. In the graphs, the $x - axis$ is time step and the $y - axis$ is distance average error (cm). The error bar in the graph represents the standard deviation. The time step is 0.5 seconds per step. According to the time step, the average distance error of $M3$ is higher than $M1$ and $M2$ in all steps in P_1. Although $M3$ has a large number of cues, the prediction accuracy is less than that of the $M1$ and $M2$. This means that even if the number of cues is large, it does not produce better prediction results.

There are two reasons why $M1$ has better performance than $M3$. First, $M3$ is more likely to be overfitted with training data than $M1$ and $M2$. Overfitting means a state in which training is too focused on training data, making it difficult to apply the model in general. The overfitted model shows excellent performance for training data but may not perform well when test data is used. In order to prevent this, in the experiment, training data and validation data are separately used during training. During training, the weights are updated only when the validation error is lower than the value of the previous epoch to prevent overfit as much as possible. Nevertheless, it can be considered that $M3$ is more likely to be overfitting because it uses more input data to predict the target than $M1$ and $M2$. This is because, in general, among DNN models having a similar structure, the more data used for predicting, the higher the probability of being overfit to training data. In this case, the structure of $M1$, $M2$ and $M3$ is almost similar. However, the amount of input is the largest in $M3$ and the lowest in $M1$. Therefore, $M3$ is more likely to be overfit than $M1$ and $M2$, and this may result in poor test performance.

The second reason is that the physical distance between T and the target robot to be predicted by model is different. This seems to be because the target robot of $M3$ has a greater distance from T than the target robot of $M1$. In the case of $M1$, there

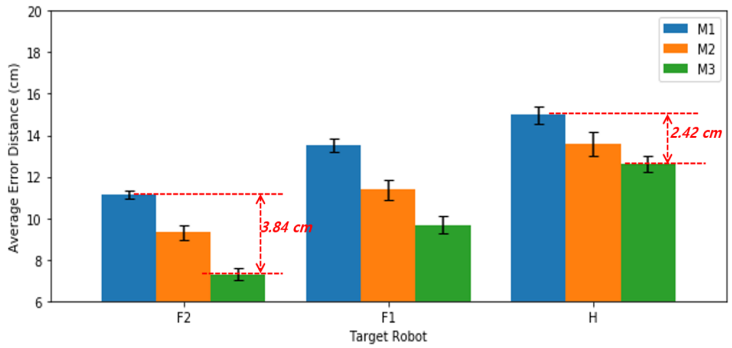
is one robot (O_1) between P_1 and T, but in the case of $M3$, there are three robots (O_1, O_2, O_3) between P_1 and T.

Also, in this experiment, it is assumed that T knows only for the positions of the observed robots and does not know the orientation, so it cannot be said that T knows perfectly about the observed robots. Therefore, even if the information of O_1, O_2, and O_3 can be known, it can be considered that the accuracy of prediction is poor because it is relatively far from T.

6.1.2 Comparison According to the Position of the Robot

Overall Performance

The prediction results for all the robots in the team-robot are shown in the graph below. The Fig. 6.3 is a graph comparing the mean error of the prediction results for H, F_1, and F_2 by models. The predictions of $M3$ for H, F_1, and F_2 are more accurate than those of $M1$ and $M2$. In addition, the results of $M3$ show better performance even if $M3$ predicts one robot farther away than that of $M1$. That is, when $M3$ predicts H and $M1$ predicts F_1, $M3$ shows better performance. Similarly, even when $M3$ predicts F_1 and $M1$ predicts F_2, $M3$ has a lower average error. Also,



p-value / Robot	M1 vs M2	M1 vs M3	M2 vs M3
F2	0.0000	0.0000	0.0000
F1	0.0001	0.0000	0.0007
H	0.0043	0.0000	0.0189

Figure 6.3: Average Prediction Error for Each Model($M1, M2, M3$) According to the Robots and P-values Between Models

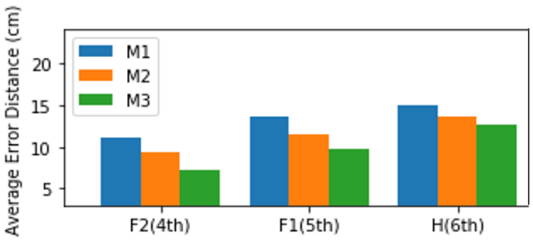
considering that the size of the robot $Thymio$(11cm), the distance difference between the predicted value of $M2$ and $M3$ for F_1 and the actual value is about the diameter of the robot.

The $p-values$ between the models are shown in the table of Fig. 6.3. All $p-values$ except $M2$ via $M3$ in H in the table are less than 0.05/3. Therefore, the assumption that the order of good performance is $M3$, $M2$, and $M1$ ($M3$ has the best performance and $M1$ has the worst performance) is reliable with a reliability of 95% or more.

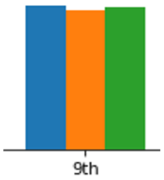
When comparing according to the number of relays, $M1$ performs better than $M3$. However, comparing the results by robot, the results of $M3$ show better performance than that of $M1$. The reason for this seems to be that the number of relay predictions is different when predicting a robot for each model. In the case of F_2, for example, in $M3$, F_2 corresponds to P_1, so $relay = 0$. On the other hand, in $M1$, F_2 is P_3 with $relay = 2$. Thus, $M1$ undergoes relay prediction twice that $M3$ does not undergo to predict F_2. When relay prediction is performed, the predicted value is used for prediction again, so that an error value increases as the number of relays increases.

In addition, the error value that is increased due to the relay is larger than the difference in the same relay for each model. When relay prediction is performed (e.g. $P_1 \rightarrow P_2$), the error increases by about 2.76cm on average. (Fig. 6.1) On the other hand, the largest difference in model prediction results for robots with the same relay is about 1.43cm (compare $M1$ and $M3$ in P_1 Fig. 6.1).

Therefore, when comparing based on the robot formation rather than the number of relays, it can be said that $M3$ shows better performance than $M1$ because there are few relays when predicting the same robot. One more thing that can be analyzed is that the greater the distance between the target and T, the smaller the difference between models. In more detail, the difference between the error of $M1$ and that



…



Robot		F2 (4^{th} from T)	F1 (5^{th} from T)	H (6^{th} from T)	
M1	E	11.14	13.51	15.00	
	ΔE		2.37	1.49	…
M2	E	9.31	11.37	13.61	
	ΔE		2.06	2.24	…
M3	E	7.30	9.68	12.58	
	ΔE		2.38	2.90	…

E : an average error distance $\Delta E : {}^{\partial E}/_{\partial Relay}$

…

(9^{th} from T)
$\tilde{E}$: 20.79 $11.14 + 5 \times 1.93$
$\tilde{E}$: 20.05 $9.30 + 5 \times \mathbf{2.15}$
$\tilde{E}$: 20.50 $\mathbf{7.30} + 5 \times \mathbf{2.64}$

Rough Estimate

Figure 6.4: Analyzing the Change Amount of Error According to the Relay to Estimate the Robot Farther Away

of $M3$ is about 3.84cm in case of F_2, but in case of H, it tends to decrease to 2.42cm. Fig.6.4 is a table and graph that summarizes the average distance error value(E:the value of $y-axis$ in the Fig.6.3) for each model and the amount of error change($\triangle E = \frac{\partial Error}{\partial Relay}$) E increases when the target is a robot far from T. $\triangle E$ means the increase in error according to the relay. The average of $\triangle E$ is 1.93 for $M1$, 2.15 for $M2$, and 2.65 for $M3$. Therefore, as relay progresses, the E of $M3$ increases more than $M1$ and $M2$. Thus, the farther away the target is from the T, the performance of $M1$ will catch up with the performance of the $M3$. Assuming that there is a robot farther away from H, we can estimate the predicted value using E and $\triangle E$. For example, when predicting a robot that is ninth from T, the results of the three models are expected to be similar. The table in the red box to the right of Fig. 6.4 represents the estimated $\tilde{E}$ values. The $\tilde{E}$ of $M1$ is 20.79cm, that of $M2$ is 20.5cm,

and that of $M3$ is 20.05cm, and the differences between $\tilde{E}$ of the three models are within 1cm.

Therefore, it can be said that all three models have similar performance. In summary, the number of cues used by the predictive model means that the effective performance difference may be shown when the length of the chain-formation is short(shorter than 8), but the performance may not be significantly affected when the length of the chain-formation is long(longer than 8). That is, when predicting a robot that is 5 robots away from T, the information of 3 robots may be significant, but when predicting a robot that is more than 5 robots away, the information of 3 robots may not be significantly different compared to the information of 1 robot.

Detailed Comparison by Time Step

Fig. 6.5 is a graph comparing the prediction error of the target robot for each model according to the time step. When comparing the results according to the time step, the average distance error is $M1 > M2 \geq M3$ even with any target in all time steps. In addition, as time passes, the difference between the values of each model tends to be larger. In particular, even considering the standard deviation, the performance difference between $M1$ and $M3$ seems significant. For H, $F1$ and $F2$, the performance is $M3 > M2 > M1$. As time passes, the error difference between models becomes larger. However, as the target robot is farther from T, the error difference between models decreases. Red arrows in Fig. 6.5 indicates the error difference between $M1$ and $M3$. The error difference between $M1$ and $M3$ in time step 10 is 8.4cm in F_2, 7.6cm in F_1, and 4.9cm in H. Through this, it can be said that the target farther from T is predicted, the narrower the differences between models are. This can be said to have a similar pattern to that mentioned in overall performance.

To analyze it in more detail, the slope of the graph can be said to be the error

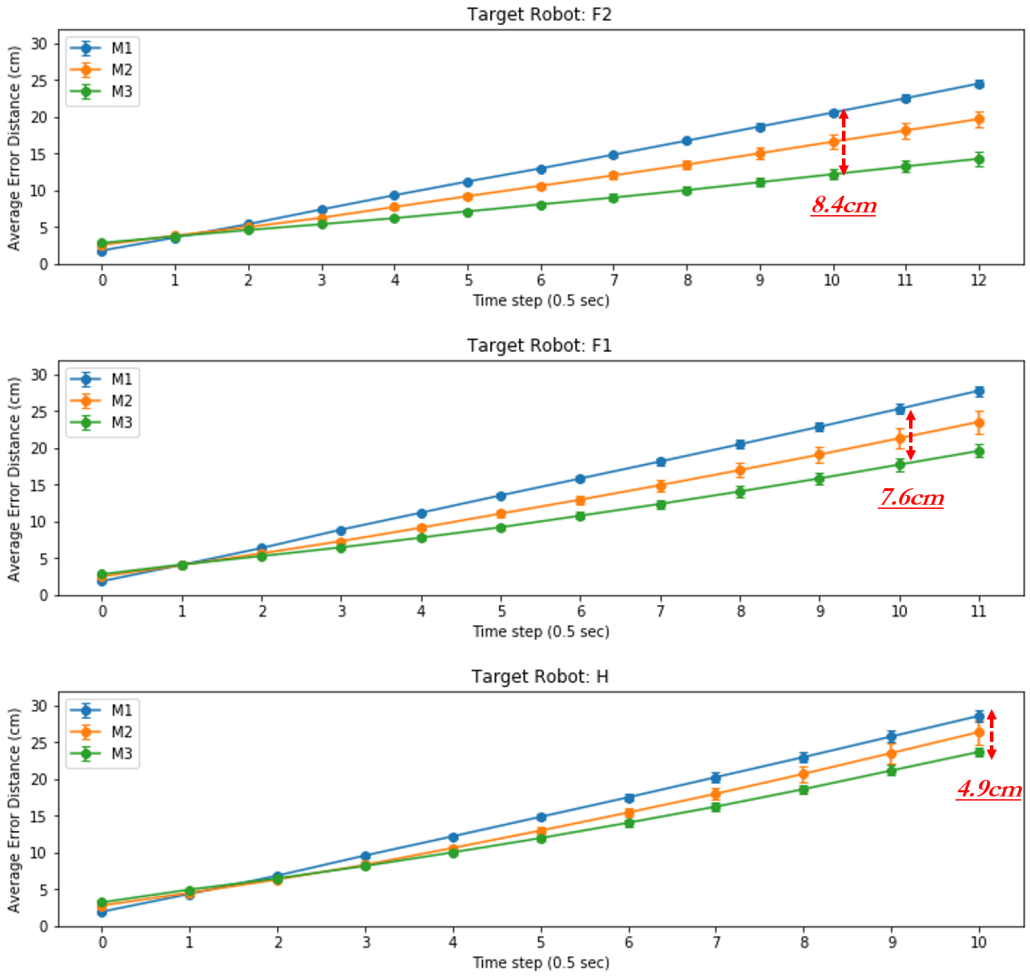


Figure 6.5: Detailed Comparison of Prediction Results for F_2,F_1 and H According to Time Step by the Models($M1$,$M2$,$M3$)

increase rate for the model according to the time step. The Fig. 6.6 shows the average of the slope of the graph(S) in the Fig. 6.5. In F_2, F_1, and H, the S is in the order of $M1 > M2 > M3$. However, $\triangle S$ is in the order of $M3 > M2 > M1$ ($\triangle S$ is amount of change in average of the slope depending on the relay $\frac{\partial Slope}{\partial Relay}$). Therefore, the farther robot from T is set a target, the smaller the difference between the S of models. For example, when experimenting with 7 robots, there are 5 robots between H and T. If you make a longer chain using more robots (e.g. using 10 robots), there will be 8 robots between H and T. If the prediction results for H in this case are estimated

for each model, the results will have slopes as shown in the Fig. 6.6. Since the initial value (at time step = 0) is similar to all three models, the similar slope means that the performance of each model is similar. Therefore, all three models will produce similar errors. This means that as the predicted target is farther from T, the effect of the number of cues on the result is reduced, and as a result, all three models will have a similar error rate.

Robot		F2 (4th from T)	F1 (5th from T)	H (6th from T)	...	(9th from T)
M1	S	1.74	2.14	2.42		$\tilde{S}$: 3.44 $1.74 + 5 \times 0.34$
	ΔS	0.4	0.28	...		
M2	S	1.32	1.79	2.24		$\tilde{S}$: 3.62 $1.32 + 5 \times 0.46$
	ΔS	0.47	0.45	...		
M3	S	0.88	1.43	1.91		$\tilde{S}$: 3.45 $0.88 + 5 \times 0.515$
	ΔS	0.55	0.48	...		

S : a slope $\Delta S : {}^{\partial S}/_{\partial Robot}$ Rough estimate

Figure 6.6: Analyzing the Change Amount of Slope According to the Relay to Estimate the Robot Farther Away

6.1.3 Qualitative Results (M1,M2,M3)

The Fig. 6.7 is the result of $M1$, $M2$, and $M3$ predicting the remote robots. Red circles are the results predicted by $M1$, yellow circles are the results predicted by $M2$, and green circles are the results predicted by $M3$. The arrow in the circle means the predicted orientation for each robot.

In all three cases (A), (B), and (C), the prediction results at the first prediction(time step t) are almost the same. Not all predicted points deviate significantly from the robot's actual position. Over time, the prediction results for each model vary slightly.

In the case of (A) and (C), $M3$ made the most accurate and $M1$ made the most

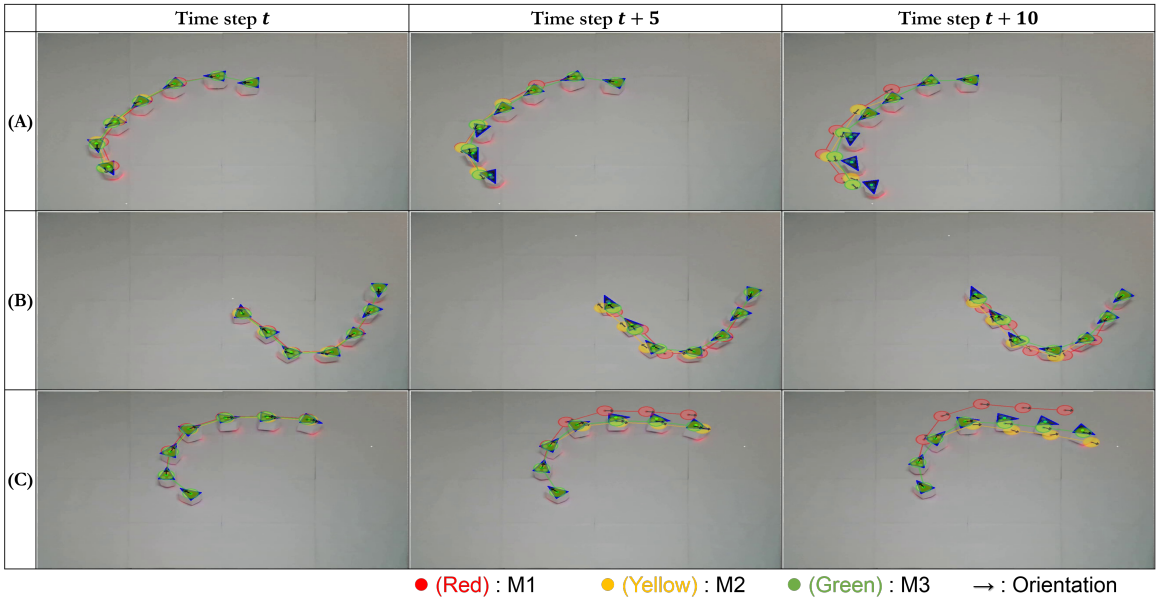


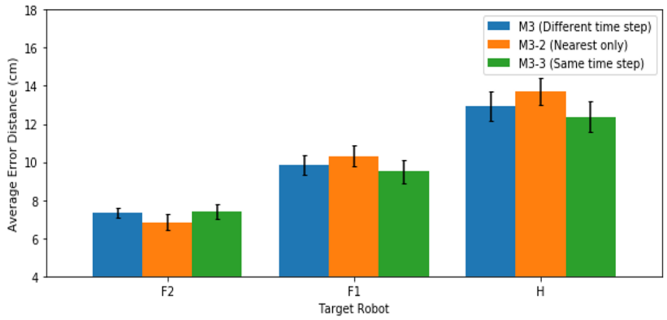
Figure 6.7: The Result of M1, M2, and M3 Predicting the Remote Robots in Frames.

inaccurate prediction. In particular, in the case of (C), the shape of the prediction result of $M1$ is similar to that of the chain, but the distance difference from the actual position is the largest among the three models. In the case of (B), the accuracy is slightly different for each model, but the prediction results of all three models are almost similar to the actual positions.

6.2 Comparison 2

6.2.1 Overall Performance

In order to compare the accuracy of prediction according to the type and composition of cues, $M3$, $M3-2$ and $M3-3$ models are created in which different types of cues are input. $M3-2$ is a model that makes inferences using only information about one observed robot (O_1) closest to the target robot, and $M3-3$ is a model that uses the same time step Observed robots' information to predict target robot.



Robot \ p-value	M3 vs M3-2	M3 vs M3-3	M3-2 vs M3-3
F2	0.0078	0.4435	0.0212
F1	0.0237	0.2072	0.0050
H	0.0125	0.1164	0.0032

Figure 6.8: Average Prediction Error for Each Model($M3, M3-2, M3-3$) According to the Robots and P-values Between Models

Fig. 6.8 shows the average error distance of the models for the robots and $p-values$ between the results of the models according to the predicted robot. In the case of F_2, $M3-2$ has better performance than $M3$ and $M3-3$. On the other hand, in case of F_1 and H, $M3$ and $M3-3$ perform better than $M3-2$. Considering the $p-value$, the $p-value$ is less than $0.05/3$ for both $M3-2$ via $M3$ and $M3-2$ via $M3-3$, so the assumptions that the performance of $M3-2$ is better than that of $M3$ and $M3-3$ in case of F_2 and the performance of $M3-2$ is worse than that of $M3$ and $M3-3$ in case of H have more than 95% reliability.

If $M3$ and $M3-3$ are compared based on the models' average error distance, $M3-3$ is better perform than $M3$ in whole cases. However, considering $p-value$, in all cases of $M3-3$ vs $M3$, since $p-value$ is greater than 0.1, it may not be accurate to analyze the correlation using only the average value. Nevertheless, the farther the target is from T, the smaller the p-value tends to be. $(0.4435 \rightarrow 0.2072 \rightarrow 0.1164)$ Therefore, if there are 8 or more robots, $M3-3$ may perform significantly better than $M3$.

It can be considered that the information at the same time step of the observed robots has more clues to predict the target than the information at different time steps. To be specific, the degree of error increase per relay ($\triangle E = \frac{\partial Error}{\partial Relay}$) according to

$\triangle E$	$F_2 \Rightarrow F_1$	$F_1 \Rightarrow H$	**Average**
M3	2.38	2.91	2.65
M3-2	3.60	3.42	**3.51**
M3-3	2.08	2.74	**2.41**

Table 6.1: Error Change Amount According to Relay by Model(M3, M3-2, M3-3)

the model is shown in the Table. 6.1. The average of $\triangle E$ is $\triangle E_{3-2} > \triangle E_3 > \triangle E_{3-3}$ with $\triangle E_3$ being 2.65cm, $\triangle E_{3-2}$ being 3.51, and $\triangle E_{3-3}$being 2.41 respectively. Based on H robot, the average error distance is the largest in $M3-2$ and the smallest in $M3-3$. Moreover, since the average of $\triangle E$ is also large in the order of $M3-2$, $M3$, $M3-3$, as the relay progresses, the difference in prediction results for each model increases and the performance will be good in the order of $M3-3$, $M3$, $M3-2$.

6.2.2 Detailed Comparison by Time Step

The Fig. 6.9 is a graph that analyzes the prediction results of each model for the target robot according to the time step. In the case of $M3-2$ in F_2, the initial prediction (prediction at time step 0) has the best performance, but at time step 12, the performance is lower than that of $M3-3$. This is because $M3-2$ has a larger error increase than other models as the time step passes. This tendency increases as the target robot is a robot farther from T. In other words, the farther the predicted target robot is from T, the greater the difference in performance for each model. The cases in F_1 and H are significantly different from those in F_2. The performance of the model is shown in order of $M3-3$, $M3$, and $M3-2$. As the time step passes, the differences in performances occur more. For example, in the case of predicting H, in step 0, the difference between $M3-2$ and $M3-3$ and $M3-3$ is 0.28cm and 0.16 cm, respectively, but in step 10, the difference is increased to 2.86cm and 4.26

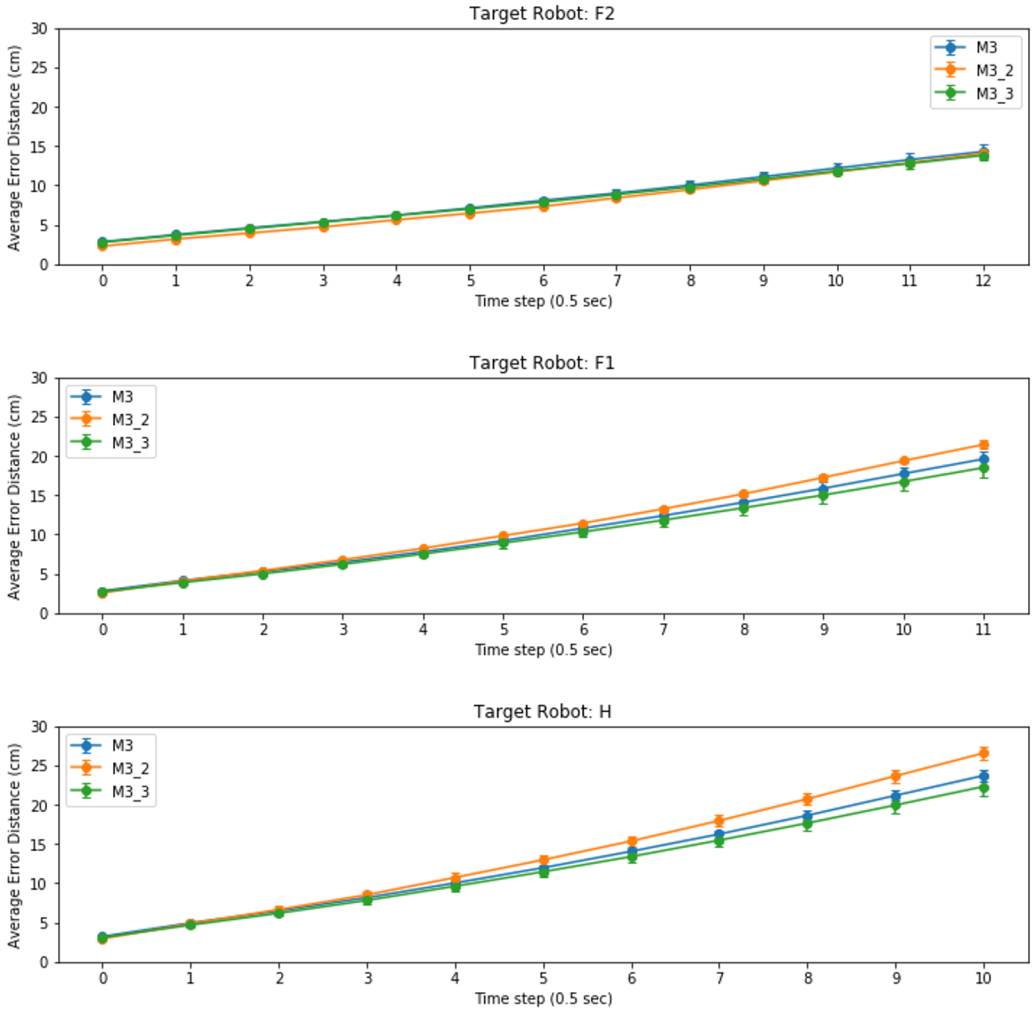


Figure 6.9: Detailed Comparison of Prediction Results for F_2,F_1 and H According to Time Step by the Models($M3$,$M3-2$,$M3-3$)

cm, respectively. To sum up, $M3-2$ has a larger error increase rate than the rest of the models as it predicts for a robot far from T. Moreover, it has largest amount of error increases as time step passes. Comparing $M3$ and $M3-3$, the information of adjacent robots of the same time-step is better than that of using different time step. To be specific, the difference between the errors increases as time passes. For H, the difference of errors is 0.12cm when time step is 0, but the difference is 1.4cm when time step is 10.

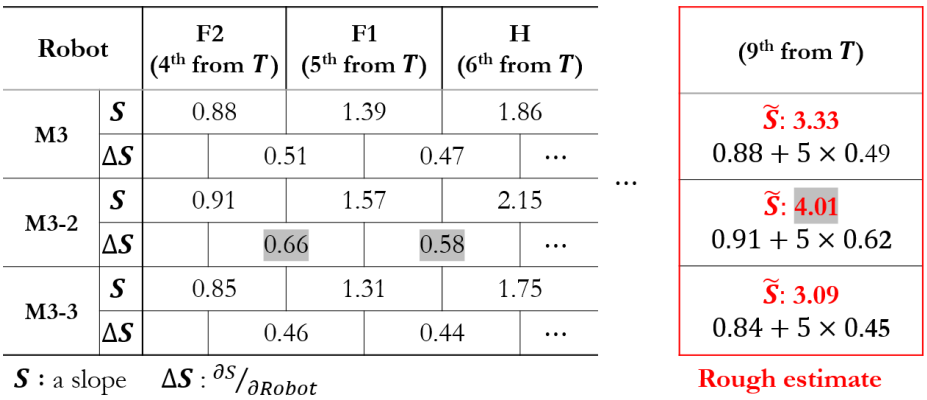


Figure 6.10: Slope Change Amount According to Relay by Model($M3, M3-2, M3-3$)

The slope(S) of the graph can be considered as the error rising rate according to the time step. Fig. 6.10 summarizes this. In F_2, F_1, and H, slope(S) is $M3-2 > M3 > M3-3$. $M3-2$ has a steeper slope than that of $M3$ and $M3-3$. In addition, the initial value (value at time step $= 0$) has a large value in the order of $M3-2$, $M3$, $M3-3$ in case of F_1 and H. Therefore, at all steps in F_1 and H, the average distance error of $M3-2$ is the largest and that of $M3-3$ is the smallest. In addition, as time passes, the difference in error between models increases. In addition, since the $\triangle S$ of $M3-2$ is the largest and that of $M3-3$ is the smallest, the farther the target is from T, the greater the difference in performances of the models. Using S and $\triangle S$, the prediction result when the length of chain-formation is longer than 7 can be estimated. The red border table on the right in Fig. 6.10 is an example. In the table, $\tilde{S}$ is an estimate of the slope based on the $\triangle S$, assuming that 10 robots are forming chain-formation. $\tilde{S}$ of $M3-2$ is the largest and $\tilde{S}$ of $M3-3$ is the smallest. Therefore, the estimation result also shows good performance in the order of $M3-3 > M3 > M3-2$.

6.2.3 Qualitative Results (M3,M3-2,M3-3)

The Fig. 6.11 is the result of $M3$, $M3-2$, and $M3-3$ predicting the remote robots. Red circles are the results predicted by $M3-2$, yellow circles are the results predicted by $M3$, and green circles are the results predicted by $M3-3$. The arrow in the circle means the predicted orientation for each robot.

In all cases, the prediction results at time step t are almost identical to the actual values. The difference in prediction results for each model appears from time step $t+5$. In particular, the results at time step $t+10$ show a large difference between $M3$, $M3-3$ and $M3-2$.

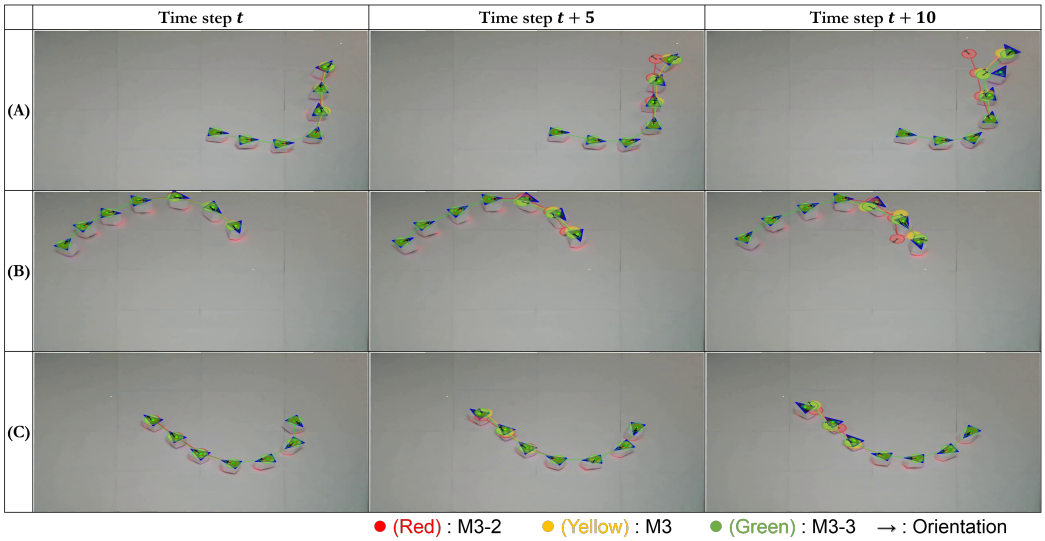


Figure 6.11: The Result of M3, M3-2, and M3-3 Predicting the Remote Robots in Frames.

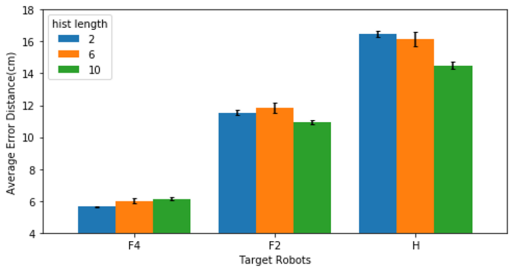
In (A) and (B), at time step $t+10$, $M3-2$ did not accurately predict the position. In addition, while $M3$ and $M3-3$ predicted the shape of the chain similar to the actual one, $M3-2$ predicted the orientation of the H as opposed to the actual one and predicted the chain shape differently. That is, the farther the target is from T, $M3-2$ tends not to make accurate predictions compared to $M3-3$ or $M3$.

6.3 Comparison 3

6.3.1 Overall Performance

In order to compare the performance difference when the history length inputted into the model is different, three models with history lengths of 10, 6 and 2 are designed and trained. Models are called $H10$, $H6$, and $H2$, respectively, depending on their history length (e.g. history length of $H10$ is 10). $H10$ is same model as $M1$. Fig. 6.12 is analyzing the average performance by models. In the case of F_4, The shorter the history length, the better the performance of the models. As the number of relays increases, the error value for each model increases. In particular, the shorter the length of $Hist$, the greater the error increasing rate as the relay progresses. In the case of F_4, the performance is good in the order of $H10 < H6 < H2$, whereas in the case of H, the performance is good in the order of $H10 > H6 > H2$.

For models with short history length, the performance is good when the target is close to the T(when the number of relays is small). Considering the case of $F4$, If the models are compared based on the $p-value$ which is 0.01 or less, $H2$ and $H6$ can be compared, and $H2$ and $H10$ can be compared. In both cases, $H2$ has a lower average distance error. Therefore, it can be said that the model with a shorter history length is more accurate.



Robot \ p-value	H2 vs H6	H2 vs H10	H6 vs H10
F4	0.003	0.000	0.130
F2	0.138	0.000	0.001
H	0.239	0.000	0.000

Figure 6.12: Average Prediction Error for Each Model($H10, H6, H2$) According to the Robots and P-values Between Models

On the other hand, for models with a long history length, the performance seems to be good when the target is far from T (when the number of relays is large). Considering the $p - value$ among the results for H, comparing $H2$ and $H10$ and comparing $H6$ and $H10$ can be said to be a more reliable comparison. In both cases, $H10$ with a long history length produces more accurate prediction results. Thus, it can be said that the larger the number of relays, the more accurate the model with a long history length.

The reason for this seems to be that the shorter the length of the history, the greater the error increase rate by the relay ($\triangle E = \frac{\partial Error}{\partial Relay}$) .Table. 6.2 shows the error increase rate by relay ($\triangle E$) for each model. The average of $\triangle E$ is smaller as the history length is longer. That is, whenever relay prediction is performed, a model with a short history length increases more errors than a model with a long history length. Therefore, the longer the history length, the less the result is affected by the relay. On the other hand, the model with a short history length is greatly affected by the relay. As a result, the performance at the initial value (F_4, relay=0) is better as the history length is shorter ($H2$), but the larger the relay (H, relay = 4), the better the model with a long history length ($H10$).

$\triangle E$	$F_4 \Rightarrow F_2$	$F_2 \Rightarrow H$	**Average**
H2	**5.88**	**4.93**	***5.41***
H6	4.93	4.33	*4.63*
H10	4.79	3.57	*4.18*

Table 6.2: Error Change Amount by Model ($H2, H6, H10$) According to the Relay

6.3.2 *Qualitative Results (H10,H6,H2)*

The Fig. 6.13 is the result of $H10$, $H6$, and $H2$ predicting the remote robots. Red circles are the results predicted by $H2$, yellow circles are the results predicted by $H6$, and green circles are the results predicted by $H10$. The arrow in the circle means the predicted orientation for each robot.

The results at time step t are similar for all three models. The predicted values do not deviate significantly from the actual positions of the robots, and the predictions are made almost accurately.

On the other hand, as time goes by, the difference between actual values and predicted values tends to increase. The prediction result for H at time step $t + 10$ seems to be $H10$ is the most accurate and $H2$ is the most inaccurate. On the other hand, the results of predicting F_4 or F_3 are similar to all three models, but $H2$ seems to be the most accurate.

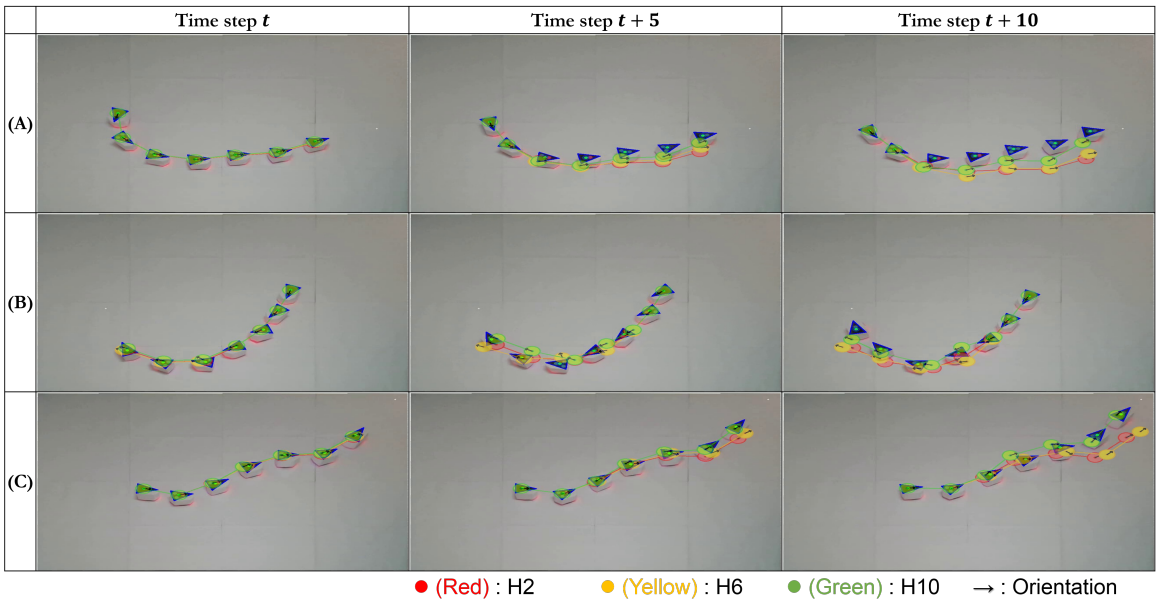


Figure 6.13: The Result of H10, H6, and H2 Predicting the Remote Robots in Frames.

Chapter 7

CONCLUSION AND FUTURE WORK

In a Multi-Robot System, there are advantages such as being able to operate decentralized when not using a global communication system. In particular, Swarm Robotics often decides the next action using only local sensing without global communication. In this situation, if robots can infer global information, they will get many advantages such as suggested in Choi *et al.* (2017). This thesis has studied how to implement physical robot team using local sensing that can infer global information using local information. Also, it is an experiment that compares the accuracy of prediction when there is a difference between the quantity and quality of local information and the query. There are two detailed goals for this.

The first target is to implement a team-robot using a real robot. To test whether global information can be inferred using local information, a physical robot platform was constructed using *Thymio*, *Raspberry pi*, and a central computer. The robots moved in one line based on the motion rule in the arena. The composition of team-robot started from 3 units and expanded to 7 units.

The second detailed goal is to infer global information using local information and compare result differences according to differences in local information. The results are as follows. First, if the number of cues is different, the model with less cues has better basic performance. On the other hand, when the comparison is made based on the absolute position of the robot without considering the relay, a model with a large number of cues shows better performance. The difference in performance according to the number of cues tends to decrease as the target is determined to be a robot far from T. Therefore, it can be said that as the size of robot team increases, the effect

of the number of cues used for prediction decreases.

Second, when the types of cues are different, the performance is better when all possible cues are used than when the cues closest to the target are used. Also, using the same time step information shows better performance than using different time step information. As the time passed, the difference between the result values tends to increase.

Finally, it is a comparison according to the history length. A model with a short history length performs better when predicting targets close to T. However, as the relay progresses, the amount of error that increases is larger in a model with a short history length. Therefore, the farther the target is from the T, the longer the history length model tends to show better performance.

As a result, the error range of the prediction model can be estimated by considering the size of the team-robot and the sensor performance of the robot using the results derived through comparison, and it is helpful in determining ideal input data.

Based on this experiment, the future work is as follows. Through the comparisons, it is possible to grasp the tendency to vary depending on the input data. Using this tendency, in Fig. 6.6 and Fig. 6.10, the predicted value for the case farther from T was estimated.However, in order to prove the estimate, it is necessary to confirm by increasing the number of robots that make up the team-robot. Therefore, I will construct a chain-formation using more robots and conduct an experiment comparing the estimated value with the actual predicted value.

In addition, using the inferred global information, it is necessary to experiment with a smart object (e.g. Tail) leading a group to a desired direction. Therefore, I will research how to create a reinforcement learning model that leads a group by using predicted results using deep learning technology by linking reinforcement learning and deep learning technology.

REFERENCES

Choi, T., S. Kang and T. P. Pavlic, "Learning local behavioral sequences to better infer non-local properties in real multi-robot systems", in "2020 IEEE International Conference on Robotics and Automation (ICRA)", (IEEE, 2020), submitted.

Choi, T., T. P. Pavlic and A. W. Richa, "Automated synthesis of scalable algorithms for inferring non-local properties to assist in multi-robot teaming", in "2017 13th IEEE Conference on Automation Science and Engineering (CASE)", pp. 1522–1527 (IEEE, 2017).

Das, B., M. S. Couceiro and P. A. Vargas, "Mrocs: A new multi-robot communication system based on passive action recognition", Robotics and autonomous systems **82**, 46–60 (2016).

Dorigo, M., E. Tuci, R. Groß, V. Trianni, T. H. Labella, S. Nouyan, C. Ampatzis, J.-L. Deneubourg, G. Baldassarre, S. Nolfi *et al.*, "The swarm-bots project", in "International Workshop on Swarm Robotics", pp. 31–44 (Springer, 2004).

Dutta, A., S. G. Chaudhuri, S. Datta and K. Mukhopadhyaya, "Circle formation by asynchronous fat robots with limited visibility", in "International Conference on Distributed Computing and Internet Technology", pp. 83–93 (Springer, 2012).

Hochreiter, S. and J. Schmidhuber, "Long short-term memory", Neural computation **9**, 8, 1735–1780 (1997).

Jevtić, A. and D. Andina de la Fuente, "Swarm intelligence and its applications in swarm robotics", (2007).

Jiang, J., J. Zhao and L.-k. Li, "Sound-based collaborative direction estimation for swarm robotic systems", Acta Automatica Sinica **33**, 4, 385 (2007).

Kelly, I. and A. Martinoli, "A scalable, on-board localisation and communication system for indoor multi-robot experiments", Sensor Review (2004).

Khaldi, B. and F. Cherif, "An overview of swarm robotics: Swarm intelligence applied to multi-robotics", International Journal of Computer Applications **126**, 2 (2015).

Ligot, A. and M. Birattari, "Simulation-only experiments to mimic the effects of the reality gap in the automatic design of robot swarms", Swarm Intelligence pp. 1–24 (2019).

Martinelli, A., F. Pont and R. Siegwart, "Multi-robot localization using relative observations", in "Proceedings of the 2005 IEEE international conference on robotics and automation", pp. 2797–2802 (IEEE, 2005).

Maxim, P. M., W. M. Spears and D. F. Spears, "Robotic chain formations", IFAC Proceedings Volumes **42**, 22, 19–24 (2009).

Nouyan, S., A. Campo and M. Dorigo, "Path formation in a robot swarm", Swarm Intelligence **2**, 1, 1–23 (2008).

Novitzky, M., C. Pippin, T. R. Collins, T. R. Balch and M. E. West, "Bio-inspired multi-robot communication through behavior recognition", in "2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)", pp. 771–776 (IEEE, 2012).

Raschka, S. and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2* (Packt Publishing Ltd, 2019).

Tan, Y. and Z.-y. Zheng, "Research advance in swarm robotics", Defence Technology **9**, 1, 18–39 (2013).