UNIVERSITY OF CALIFORNIA
RIVERSIDE

Experimental Validation of Game Theory Based Coverage Control

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Jiahui Lu

December 2016

Thesis Committee:
Dr. Wei Ren, Chairperson
Dr. Ertem Tuncel
Dr. Daniel Wong

The Thesis of Jiahui Lu is approved:

Committee Chairperson

University of California, Riverside

To my parents

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Current scientific research, especially in high-end fields, has been focusing on theoretical demonstration because the lack of proper platforms and techniques. Researchers usually prove their theory through simulations. However, even the most perfect simulation may have some machenism that cannot be completely simulated. This calls for experiment validation on physical robots. Based on ARIA library which is written by the robot's manufacturer, ready-to-use programs are written to make experiments easy to carry out.

1.2 Literature Review

Improvement in wireless network techniques and expense reduction in individual sensing devices have created a great foundation for Mobile Sensor Networks(MSNs). Mobile robots with all kinds of sensors on it are utilized to validate various distributed control algorithm. In [1], distributed consensus algorithm such as rendezvous and axial alignment are demonstrated with physical experiments. In its platfrom, an overhead camera is used to serve as a pseudo-GPS which determines the position and orientation of each robot. The research done in [2] doesn't need overhead camera because the robot they use has relatively high precise encoder and it can get position and orien-

tation based on its data. Physical experiments has also been done to demonstrate algorithms like distributed containment control for double-integrator dynamics in the presence of both stationary and dynamic leader. An Amigobot and Pioneer 3-DX based multi-robot platform is used in [3] for exploration of formation control strategy. Becasue only local information is used, this experiment disallows the use of information obtained from certain number of groups to emulate limited inner-robot information exchange.

Along with distributed control algorithms, there are other experiments that have been demonstrated using physical experiments. [4] presents an experimental implementation and validation of a localization system based on a heterogeneous sensor network. It demonstrates the combination of ultrasound and web-camera is able to provide multitude of services with guaranteed quality with intelligent sensor fushion scheme. [5] provides a Cooperative LOcalization with Quality of estimation(CLOQ) algorithm which improved the localization performance for entire wireless sensor network. It also experimentally validates the CLOQ algorithm based on Received Signal Strength Indication(RSSI) measurement.

1.3 Main Contribution

This thesis studies the coverage problem in an unknown environment. It can be modelled as one of the resource allocation problems which entails a collection of dispersed interacting components seeking to optimize a global collective objective through local decision making. Limited communication capabilities, local and dynamic information, faulty components, and an uncertain environment make the problem very complicated. It is not feasible to pass all information to a command center to process. Even if it were possible, the complexity of the overall system makes the problem of constructing a centralized optimal policy intractable. Hence, a distributed control algorithm of MSNs is designed in [6] .

The implementation of the above algorithm is the main contribution of this thesis. First, a simulation will be conducted using MobileSim which makes debugging easier and faster. After validated with simulation, experiments on physical robots will be performed. The communication

structure of this system is client-server. Each robot will run a server program on its on-board PC while multiple client programs will be initiated on peronal laptop. Clients and servers are connected through TCP port of robots' on-board PCs. The robot control system including robot manipulation and network is developed using Advanced Robot Interface for Application(ARIA) which provides various classes and funtions in C++ language.

1.4 Outline for Chapters

Chapter 2 covers the theoretical knowledge of the main algorithm this thesis wants to validate from [6]. It goes through all the terms and explained how they are derived.

Chapter 3 covers the hardware and software used by the experiments. It includes everything from actual part to the libraries used. In addition, the architecture of the brief control system is explained in depth.

Chapter 4 presents the settings and results of both MobileSim simulations and physical experiments. Experimental data is intepreted as well in this chapter.

Chapter 5 goes over several possible improvements to the experiment, potential future work and a conclusion.

Chapter 2

Game Theory Based Control Algorithm for Sensor Coverage

This part is work of [6]. I include it for completeness.

2.1 Problem Definition

An MSN is established by a distributed collect of agents where each of them has sensing, computation,communication and moving ability. In our problem, a limited number of agents are randomly deployed in a task area. The ultimate goal is to cover the most worthy locations with minimum energy consumption.

In reality, the worth of a place can be interpreted in many ways. For example, the probability of finding a target by sensors on that location. However, sensors will not have prior knowledge about the worth of each location and can only determine after sensing. To be more specific, we will be solving a coverage optimization problem in an unknown environment. To tackle this, we will formulate it as a non-cooperative game where each sensor communicates with neighbors within sensing radius.

A convex two-dimentional mission space is considered. The whole space is discretized into squared lattices. Each lattice has unit dimension and is labelled with the coordinate of its center

$q = (q_x, q_y)$.The collection of all squares of the lattice is denoted by Q. A numerical variable $f_q \geq 0$ is assigend to the worth or the probability of the occurence of an event in each square. f_q is assumed to be stationary.

There are N mobile sensor agents deployed in the mission space and their locations can be denoted by $a_i^p(t) \triangleq (x_i(t), y_i(t)) \in Q$. The sensing region of agent i is modelled as a disc with center $a_i^p(t)$ and the radius $a_i^r(t)$, where $a_i^r(t)$ is chosen from a discrete set with the minimum and the maximum equal to r_{min} and r_{max}. All agents are assumed to have the same r_{min} and r_{max} while they can have different discrete sets. The squares covered by agent i, $S(a_i(t))$, is a function of the agent's action $a_i(t)$ where $a_i(t) := (a_i^p(t), a_i^r(t)) \in A_i$ and A_i is available action set for agent i. The action profile of all agents is denoted by $a(t) = (a_1(t), .., a_N(t)) \in A := \prod_{i=1}^{N} A_i$. At each time t, an agent can only move to the squares around it. We use $C_i(a_i(t-1))$ to denote the available actions for agent i at time step t. Each agent is only able to communicate with its neighbors to exchange information. The set of neighbors of agent i is given by $N_i^{comm}(a(t)) := \{j = 1, ..., N | (x_i - x_j)^2 + (y_i - y_j)^2 \leq (R_i^{comm})^2\}$, where R_i^{comm} is the communication range of agent i.

2.2 Game Design

In this section, a game will be designed to drive agents to better locations with larger total worth while still considering the energy consumption.

2.2.1 Utility Design

In order to fully make use of energy spent on communication and sensing, we assume $R_i^{comm} = 2r_{max}$ which means whenever two agents have intersection in their sensing area, they can communicate with each other. In addition, the only information that will be shared with neighbors is its own action $a_i(t)$ which further reduces energy consumption.

We then consider the energy consumption caused by sensing. There is a trade off between power

usage and the size of covered area. Agent i's energy consumption on sensing is defined

$$E_i^{sense} = K_i(a_i^r(t))^2$$

where $K_i > 0$ is a coefficient. $a_i^r(t)$ is used as an optimization variable. In [7], a coverage algorithm for vision-based sensors is proposed to turn off the sensors with overlapping fields of view. In our proposed method, if we let $r_{min} = 0$, which means agent i can choose $a_i^r = 0$ when the agent cannot improve the sensing performance.

We finally consider the energy consumption caused by movement. This is defined as

$$E_i^{move} = K_i'(|a_i^p(t) - z_i(t)|)$$

where $K_i' > 0$ is a coefficient and $z_i(t) \triangleq a_i^p(t-1)$ is the previous location of agent i. Here the agent's previous location is its current state and we can see the energy consumption caused by movement is a funtion of current state. This is the reason that a state-based potential game will be designed for this game.

We now proceed to formulate our coverage optimization problem. A utility function U_i is designed for each agent that aims to capture the trade off between the worth of covered area and the energy consumption by agent i.

$$\begin{aligned} U_i(a(t), z_i(t)) =& F(a_i(t), a_{-i}(t)) - F(a_i^0(t), a_{-i}(t)) \\ & - K_i(a_i^r(t))^2 - K_i'(|a_o^p(t) - z_i(t)|) \end{aligned} \tag{2.1}$$

where

$$F(a(t)) = \sum_{q \in \bigcup_{i=1}^{N} S(a_i(t)) \cap Q} f_q$$

denotes the worth of the covered area by the agents. $a_i^0(t)$ is the null action for agent i and $a_{-i}(t)$ is the action of all agents other than agent i. As a result, $F(a_i(t), a_{-i}(t)) - F(a_i^0(t), a_{-i}(t))$ is agent i's marginal contribution to sense the covered area. From eq. 2.1 , we can easily find out that the

utility function of agent i only depends on the actions of $\{i\} \cup N_i^{sens}(a(t))$, where $N_i^{sens}(a(t))$ is the set of all agents that have an intersection with that of agent i. Covered regions of agents that have no intersection with agent i are eliminated. As a result, U_i is local.

After introducing all the ingredients, a state-based game will be intrduced in the following lemma.

Lemma 2.1 *The coverage state-based game* $\upsilon :=< N, A, U_{cov} >$*, where* $U_{cov} = \{U_j, j = 1, ..., N\}$ *is an exact state-based potential game with the potential function*

$$\begin{aligned}\Phi(a(t), z(t)) = & \sum_{N}^{j=1}(F(a_j(t), a_{-j}(t)) - F(a_j^0(t), a_{-j}(t))) \\ & - \sum_{j=1}^{N} K_j(a_j^r(t))^2 - \sum_{j=1}^{N} K_j'(|a_j^p(t) - z_j(t)|)\end{aligned}$$

where $z(t) = (z_1(t), ..., z_N(t))$ *is the current state of the game.*

[6] provided proof of this game.

2.2.2 Reinforcement Learning

In order to converge to Nash equilibrium while maximize agent i's own utility function, game theoretic reinforcement learning is designed here[8][9].The binary log-linear learning method has been analyzed in [10]. Theorum 5.1 in [10] shows that a potential game will converge to stochastically stable actions. These actions are set of potential maximizers if all agents adhere to the binary log-linear learning where the following assumptions should be satified on the agents' available sets.

1. Feasibility: For any agent $i = 1, .., N$ and any action pair $a_i(0), a_i(m) \in A_i$.There exists a sequence of actions from $a_i(0)$ to $a_i(m)$ satisfying $a_i(t) \in C_i(a_i(t-1))$ for all $t \in 1, 2, ..., m$.

2. Reversibility: For any agent $i = 1, ..., N$ and any action pair $a_i', a_i'' \in A_i, a_i' \in C_i(a_i'') \longleftrightarrow a_i'' \in C_i(a_i')$.

It's easy to check that these assumptions are satisfied. We will use binary log-linear learning method to update each agent's action.

At each time t, only one agent will be randomly chosen and allowed to alter its action while the others will repeat their actions. The chosen agent i will choose a trial action $a_i(t)$ uniformly randomly from the available action set $C_i(a_i(t-1))$. It then calcultes the utility function for this trial action and randomizes its action according to

$$\begin{aligned} P_i^{a_i(t-1)}(t) &= \frac{exp(\frac{1}{\tau}U_i(a(t-1), z(t-1)))}{exp(\frac{1}{\tau}U_i(a(t-1), z(t-1))) + exp(\frac{1}{\tau}U(a_i'(t), a_{-i}(t-1), z(t)))} \\ P_i^{a_i'(t)}(t) &= \frac{exp(\frac{1}{\tau}U(a_i'(t), a_{-i}(t-1), z(t)))}{exp(\frac{1}{\tau}U_i(a(t-1), z(t-1))) + exp(\frac{1}{\tau}U(a_i'(t), a_{-i}(t-1), z(t)))} \end{aligned} \tag{2.2}$$

where $P_i^{a_i(t)}(t) = 0, \forall a_i(t) \neq a_i'(t), a_i(t-1)$. It denotes the probability of choosing action $a_i(t)$ at time t. For $\tau \to \infty$, the learning algorithm will choose the action $a_i(t-1)$ or $a_i'(t)$ with an equal probability while for $\tau \to 0$, it will choose the action which has the larger utility fuction between $a_i(t-1)$ or $a_i'(t)$. In other words, an agent is randomly chosen and it will choose between alternative action $a_i'(t)$ and previous action $a_i(t-1)$ randomly. The alternative action $a_i'(t)$ is randomly selected from the set $C_i(a_i(t-1))$.

2.3 Gaussian Mixture Model Estimation

As stated in 2.2.2, agent i needs to decide whether to go to alternative action based on the utility function values of both current action and alternative action. Since at that time, agent i hasn't been there, it needs to estimate the worth of alternative action's covered area. We will use Gaussian Mixture Model(GMM) as an estimation model of the worth of the area. In our setting, agents will keep the worth of their sensed locations in their memories. The GMM is a parametric probability density function represented as a weighted sum of Gaussian component densities defined as

$$\sum_{k=1}^{M} w_k g(x|\mu_k, \sigma_k)$$

and

$$g(x|\mu_i, \sigma_i) = \frac{1}{2\pi|\Sigma_i|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i))$$

where x is a 2-dimentional location data vector, M is the number of Gaussian components in the GMM. w_i is the i_{th} weighting coefficient and $g(x|\mu_i, \Sigma_i)$ is a 2-variate Gaussian distribution with mean vector μ_i and covariance matrix Σ_i. The weights have to satisfy the constraint that $\sum_{k=1}^{M} w_k = 1$. The reason to use GMM is it is one of the most frequently observed distributions and is a starting point for modelling many natural processes.

In this paper, the GMM is parametrized by $\lambda = \{w_j, \mu_j, \sigma_j | j = 1, 2, ..., M\}$ where w_j, μ_j, and σ_j are the weighting coefficient, mean and covariance corresponding to the j_{th} Gaussian of GMM. One of the most accurate and well-established method is the ML estimation. Agent i has the observation O_t^i at time step t.The sequence of these observations $O^i = \{O_1^i, O_2^i, ..., O_T^i\}$ can be used as training vector with which we wish to find the estimate of λ^i that maximizes the probability of the occurence of the observation sequence vector denoted by $p(O^i|\lambda^i)$. The ML estimation can be written as

$$\max_{\hat{\lambda}^i} p(O^i|\hat{\lambda^i}) = \max_{\hat{\lambda}^i} \prod_{t=1}^{T} p(O_t^i|\hat{\lambda}^i)$$

This is a non-linear function of parameter $\hat{\lambda}^i$ and direct maximization is not possible, the expectation maximization (EM) has been introduced in [11] to obtain an iterative solution. Using the same EM approach, the GMM parameters can be estimated as

$$\begin{aligned}
\hat{w}_j^i &= \frac{1}{T} \sum_{t=1}^{T} Pr^i(j|O_t^i, \hat{\lambda}^i) \\
\hat{\mu}_j^i &= \frac{\sum_{t=1}^{T} Pr^i(j|O_t^i, \hat{\lambda}^i) O_t^i}{\sum_{t=1}^{T} Pr^i(j|O_t^i, \hat{\lambda}^i)} \\
{\hat{\sigma}_j^i}^2 &= \frac{\sum_{t=1}^{T} Pr^i(j|O_t^i, \hat{\lambda}^i)(O_t^i - \hat{\mu}_j^i)(O_t^i - \hat{\mu}_j^i)^T}{\sum_{t=1}^{T} Pr^i(j|O_t^i, \hat{\lambda}^i)} \\
Pr^i(j|O_t^i, \hat{\lambda}^i) &= \frac{\hat{w}_j^i g(O_t^i|\hat{\mu}_j^i, \hat{\sigma}_j^i)}{\sum_{k=1}^{M} \hat{w}_k^i g(O_t^i|\hat{\mu}_j^i, \hat{\sigma}_j^i)}
\end{aligned} \tag{2.3}$$

where $Pr^i(j|O_t^i, \hat{\lambda}^i)$ is a posteriori probability of the observation O_t^i for the j_{th} Gaussian of agent i.

One problem of this algorithm is that the EM method optimizes the likelihood of the parameters given the observed location without taking the worth of that location into account. The same problem has been pointed out in the image processing literature [12]. We use a modified version of the

solution introduced in [12] to repeat the ML algorithm m times in worthy locations with m chosen as

$$m = \begin{cases} 1 + \gamma * round(\frac{f_q}{f_{mode}}) & f_q \geq f_{mode} \\ 1 & otherwise \end{cases} \tag{2.4}$$

where f_{mode} and γ are a threshold and a correction factor. It is apparent that if the correction factor,γ, is increased, a region with a higher worth will be repeated more often.

In our solution, the agents use the estimated parameter $\hat{\lambda}^i$ to calculate the estimation of f_q at each iteration which is used to evaluate the utility function. Although f_q is asumed to be stationary, the estimate of f_q is no longer stationary. However, a standard assumption in game theory is that the game parameters are stationary. In [13],the case of slow variations of the game parameters is investigated. Corollary 4 in [13] shows that using the binary log-linear learning the probability of converging to Nash equilibrium will be greater than $1 - \delta_1$ if the following is satisfied:

$$|\lambda(t+1) - \lambda(t)| < \delta_2$$

$$|\hat{\lambda}^i - \lambda(t)| < \delta_3$$

where δ_2 and δ_3 are the bound on the changing rate of λ and the bound on estimation error of λ. Under the assumption of a stationary environment in Section II we will have $\delta_2 = 0$. In addition, the agents' estimated parameters $\hat{\lambda}^i(t)$ converge to $\lambda(t)$, which implies that the estimation error decreases. Hence there exists a bound δ_3 on the estimation error. Thus the conditions are satisfied and the agents will converge to a Nash equilirium stochatically.

2.4 Mutual Information

In this section, we add a mutual information term to each agent's utility function in order to find an action and its observation is more informative. In order to select informative observation, the entropy criterion has been used in information theory and applied mathematics contexts [14]. In information theory, the conditional entropy quantifies the amount of information needed to de-

scribe the outcome of a random variable Y given the value of another variable X, which is written as $H(Y|X)$. Here our goal is to reduce the uncertainty of the unobserved area by finding more informative locations for future observations. In [15], the mutual information criterion has been proposed for observation selections. It is shown that maximizing the mutual information which represents independence between an observed and an unobserved area is more effective than the conditional entropy to reduce the uncertainty.

The objective function can be written as

$$I(X_{O^i} : X_{Q\setminus O^i}) = H(X_{Q\setminus O^i}) - H(X_{Q\setminus O^i}|X_{O^i}),$$

where I, X_{O^i} and $X_{Q\setminus O^i}$ are the mutual information, and the observed and unobserved variables corresponding to agent i, respectively. Fortunately, for Gaussian processes there exists a closed form to compute the conditional entropy which is given by

$$H(X_{Q\setminus O^i}|X_{O^i}) = \frac{1}{2}\log(2\pi e\sigma^2_{X_{Q\setminus O^i}|X_{O^i}}),$$

where e is the Euler number and $\sigma^2_{X_{Q\setminus O^i}|X_{O^i}}$ is the covariance of the conditional distribution of $X_{Q\setminus O^i}$ given X_{O^i}. The covariance $\sigma^2_{X_{Q\setminus O^i}|X_{O^i}}$ is calculated as

$$\begin{aligned}\sigma^2_{X_{Q\setminus O^i}|X_{O^i}} &= K(X_{Q\setminus O^i}, X_{Q\setminus O^i}) \\ &\quad - \Sigma_{X_{Q\setminus O^i}X_{O^i}}\Sigma^{-1}_{X_{O^i}X_{O^i}}\Sigma_{X_{O^i}X_{Q\setminus O^i}}\end{aligned}$$

where $\Sigma_{X_{Q\setminus O^i}X_{O^i}}$ is a covariance matrix each of whose entries is a function of the kernel function $K(a,b)$ for $a \in X_{Q\setminus O^i}, b \in X_{O^i}$, and $\Sigma_{X_{Q\setminus O^i}X_{O^i}} = \Sigma^T_{X_{Q\setminus O^i}X_{O^i}}$. The kernel function determines the correlation of the observations and can be defined using different functions. One of the most frequently used kernel functions is $K(a,b) = exp(-\frac{\|a-b\|_2^2}{h^2})$, where $\|a-b\|_2$ is the distance between the locations a and b and h is a constant. As can be seen, the covariance does not depend on the observed values, so the mutual information can be calculated ahead of an agent's new action.

This is important because we want to compare the utility value of current and trial action. The latter one happens before agent's actual movement. By adding the mutual inforamtion term to the utility function, we get a modified game which was proved to be a state-based potential game as in [6]. We then define a new utility function

$$\begin{aligned} U_i(a(t), z(t)) &= F(a_i(t), a_{-i}(t)) - F(a_i^0(t), a_{-i}(t)) \\ &- K_i(a_i^r(t))^2 - K_i'(|a_i^p(t) - z_i(t)|) \\ &+ \eta(t)(H(X_{Q\setminus O^i}) - H(X_{Q\setminus O^i}|X_{O^i})). \end{aligned} \tag{2.5}$$

where $\eta(t)$ is a time varying coefficient which adjusts the importance of the mutual information term in comparison with the sensing optimization part. The selection of $\eta(t)$ follows the following principle. Starting the search, the agents do not have a good estimate of the area. Gathering proper data serves as an important role. However, by having a more accurate estimate, the weight of the sensing part needs to be increased in order to put more effort on finding the best configuration of the MSN.

Chapter 3

Hardware and Software

3.1 Pioneer 3-AT

Pioneer 3-AT is a versatile mobile robot designed by The Adept MobileRobots. It has an on-board computer which opens the way for Ethernet-based communication, laser, vision processing and other autonomous functions. Its precise encoder is the main reason to choose it as this experiment's platform. Differential drive kinematics is used in Pioneer 3-AT. There are serval ways to drive it with the help of ARIA library. Setting linear and angular velocity, setting velocities of two sides of wheels respectively or setting the goal position in global frame directly. Among all three ways, I used the first way to keep the compatibility with iRobot Create Roomba, as we may need to do experiments with heterogeneous robots later.

3.1.1 Configurations

Pioneer 3-AT:

- 44.2368MHZ Renesas SH2 32-bit RISC microprocessor.

  This handles low level details of mobile robotics, including maintaining the robot's drive speed and heading, as well as acquiring and pre-processing sensor readings.

- On-board computer

- 2x Intel(R) Core(TM)2 Duo CPU P8400 @ 2.26GHz
- 2021MB DDR3 Memory

  This runs a server on Windows 7 home edition. It communicates with the robot's microcontroller through its HOST serial port and the dedicated serial port COM1 under Windows.

3.1.2 Specifications

- Velocity range: -1200 to 1200 mm/s
- Rotational velocity range: -300 to 300 degree/s

3.2 Camera

The camera is only used to take videos of this experiment. Because Pioneer 3-AT has relatively high precise encoder, it's unnecessary to utilize camera data to correct poses of robots.

3.2.1 Configurations

- Samsung Galaxy Note 5 front camera
  - 5.0 megapixel sensor
  - 1920x1080 resolution

3.3 Router

This is the pivot of communication. No special configuration is needed. The only thing we need to do is to connect all the devices to it. It will automatically assign IP addresses through which we can establish connections between laptop computer and on-board PCs.

- Linksys N300 WIFI Rounter E1200

3.4 Computer

This is the central ”brain” of the experiment. It will collect all the data including current postions and sening radius of robots and store worth of sensed locations. Based on these data, it will calculate next goal postion of selected robot. Control command then will be sent to that robot's server.

3.4.1 Configurations

- CPU: Intel Core i5-5257U 2.7GHz
- RAM: 8GB

3.5 Softwares

The central computer runs on Windows 7 professional 64-bit, trial version.

3.5.1 Programs

- MobileSim - A simulator using the model of physical robot. This program is linked with ARIA and the simulation is conducted using C++ code.
- MATLAB - Used to extract data from output files and plot trajectories of all robots.
- Microsoft Visual Studio 2013 professional - Used to edit and compile codes.
- Parrallel Desktop - Used to boot Windows 7 on OS X 10.11

3.5.2 Libraries

- Advanced Robot Interface for Applications(ARIA) - Provides functions to build up network and to drive Pioneers.
- Eigen - Library which defines Matrix types and performs linear algebraic oprations with ease.
- GNU Scientific Library 1.8 - Library used to generate random number and to simulate bivariate normal distribution.

Figure 3.1: A picture of Pioneers

3.5.3 Robot Control System and Architecture

The physical robot experiments are based on three Pioneer 3-ATs from Adept Mobilerobots that are shown in Fig. 3.1. A top level diagram of the multi-agent control system architecture is shown in Fig. 3.2. Each robot is equipped with an on-board PC on which a server program is running. The on-board PC is connected to the robot internally. Three client programs run on one laptop, which makes the communication between clients easier. Although information of all actions is accessible, we emulate the distributed control scheme by using limited inner-robot information exchange. As it is shown by dashed arrows in Fig. 3.2, the robots are not allowed to use the actions of the robots that are not in their communication neighbor set. This means client i does not have access to the action of client j, if the distance between robots i and j is larger than R^{comm}. This enables us to test distributed control algorithms that involve only local information. The laptop and all three robots join the same WIFI network, where the communication between clients and servers is based on TCP.

A client program runs on the laptop, which keeps requesting a robot's action (see appendix A.2 Line 9-18, 25-36) and calculating the new action (see appendix A.3 Line185-363) which implements (2.2), (2.3) and (2.4). The on-board PC is responsible for high-level control. It runs a server that keeps monitoring on a specified TCP port. According to the received command, it will choose to send the robot's position to a client or to receive a new action from the client(see appendix B.2 Line 7-17) . The Pioneer 3-AT also has a microcontroller which is designed to perform low-level control. It maintains support for all sensing and actuation features of the robots. A high-precision

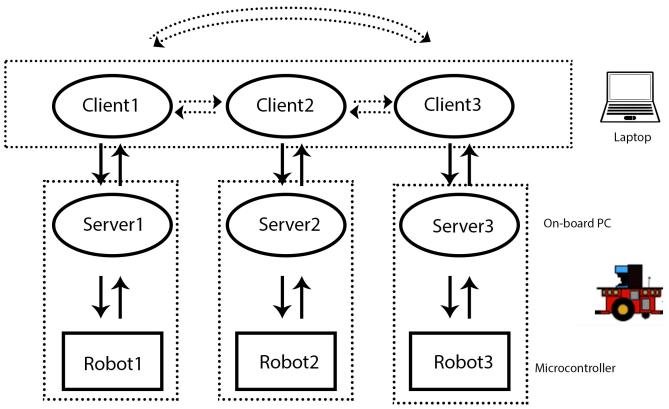


Figure 3.2: The architecture of platform

encoder/decoder will be utilized where there is no need for correction on the calculation of the distance traveled nor angle turned. Each server periodically receives the current pose, (position and orientation), from the microcontroller. We set the delay of server at 100 ms deliberately since the pose of robot stored in server will be out of date if the delay is larger and there is no need for it to be zero delay as the client spends some time to calculate. By using a feedback control(see appendix B.3 Line 120-180), the server generates a new command and sends it to the microcontroller in order to move the robot into its desired position. A deadband of 2 degrees and 2.236 cm are incorporated to prevent oscillation.

Chapter 4

Tests and Results

Before real experiments, simulations are usually done first. Because all processes are simulated on computer, each trial costs less time and it's easier to make changes to configuration and parameters. A Matlab simulation has been done in [6]. In this thesis, MobileSim simulation is carried out first and after it works well, physical experiments with Pioneer 3-AT are conducted. Both MobileSim simulation and physical experiments implement C++ code. The only difference between MobileSim simulations and physical experiments is that three simulated robots will be connect through local TCP ports while real robots are connected through TCP port of its own.

4.1 Results of MobileSim

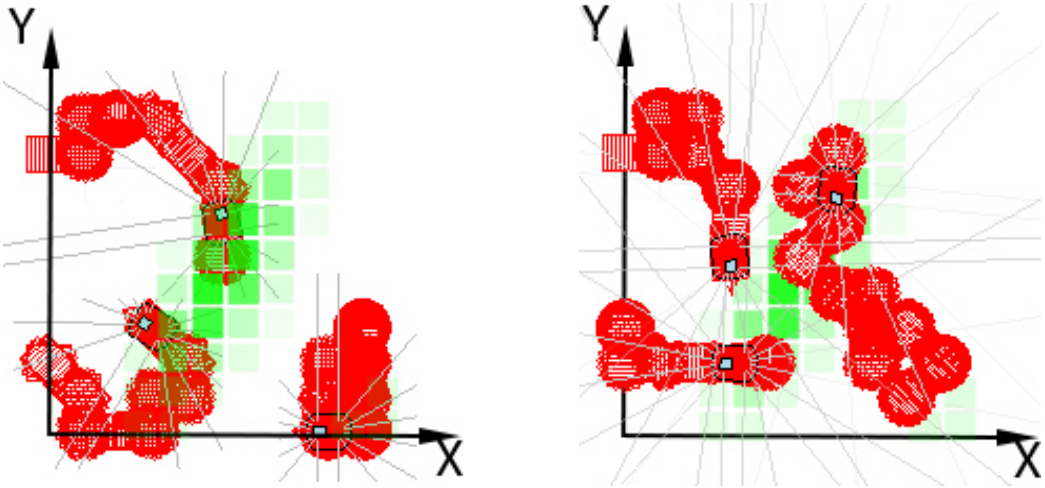
The algorithm stated in chapter 2 is first simulated using MobileSim. A 5×5 meters area is created, which is discretized into a 10×10 squared lattice. Fig. 4.2 shows the distribution of the worth, where a worthy area is assumed on the diagonal. We also deliberately introduce a worthy area on the corner. This will illustrate the advantage of using the mutual information term to escape the suboptimal equilibria. To show how the proposed algorithms perform in real applications, an obstacle is also introduced in the coverage area, where the obstacle is shown by red squares in Fig. 4.2. A group of three robots ($N = 3$) are deployed in this area. The robots

are initialized at $\begin{pmatrix}1\\3\end{pmatrix}$, $\begin{pmatrix}1\\9\end{pmatrix}$, and $\begin{pmatrix}10\\3\end{pmatrix}$, respectively, where their initial positions are shown by green squares in Fig. 4.2. Each robot can choose its a_i^r from the set $\{0, 1, 2\}$, which means $r_{\text{min}} = 0, r_{\text{max}} = 2$, and $R^{\text{comm}} = 2r_{\text{max}} = 4$. We let $M = 1$ and use the Gaussian distribution as an estimation model. The agents employ the proposed approach using (2.2), (2.3) and (2.4) with the utility function (2.5). The experiment parameters are chosen as $K_i = 2 \times 10^{-4}, K_i' = 2.5 \times 10^{-2}, i = 1, 2, .., N, \tau = 5 \times 10^{-3}, f_{\text{mode}} = 10^{-3}$, and $\gamma = 1$. After running 400 steps for both cases(with and without mutual information), we get Fig. 4.1a and Fig. 4.1b, showing the trajectories of three robots as red prints. Different percentage of opacity indicates different worth of each location. As can be inferred from Fig. 4.2, the lighter it is, the less worth it bears. Some explanations are more understandable when we look at the matlab plot based on physical experiments' data and we will present them in the next section. What is worth mentioning here is the choice of η, the weight of mutual information term. As stated in chapter 2, the portion of mutual information among utility function should decrease as agents need to put more effort on optimization of their deployment rather than gathering information. The speed of decreasing matters as well. Compared with $\frac{0.01t+5}{t+10}$, if η decreases faster, robot 3 which is initialized at the corner will never move out of the corner. If η is large and decreases too slow, the system will move a lot which obviously violates our original intension.

4.2 Results of Physical Experiments

The experiment area is prepared as shown in Fig. 4.3. A coordination is marked on the ground. Because the radius of Pioneer 3-AT is approximate 0.5 meter, the unit is chosen as 500 millimeters. Different percentage of opacity of green sqaures and red squares are also added to Fig.4.3.

In our first experiment the utility function (2.5) has been used with $\eta = 0$, where the robots maximize the worth of the covered area while considering the energy consumption. Fig. 4.4 shows the initial positions, the trajectories and the final positions of the robots. Explanation of green colored squares remains the same and the obstacles are shown by red squares. As we can see,



(a) Trajectories of robots using the utility function (2.5) and for $\eta = 0$ (without mutual information)

(b) Trajectories of robots using the utility function (2.5) and for $\eta = \frac{0.01t+5}{t+10}$ (with mutual information).

Figure 4.1: MobileSim results

robots 1 and 2 move toward better locations, which can cover more worthy areas. After 400 steps, their final sensing regions are indicated by yellow circles. Robot 3 is initialized at the corner, where it can sense some worthy areas. As it is shown in Fig. 4.4, robot 3 decides to stay there because the neighborhood areas have lower worth than its current position and the robot has no knowledge of more worthy areas far from its initial position.

In our next experiment, the utility function (2.5) has been used with $\eta = \frac{0.01t+5}{t+10}$. Here the mutual information term in the utility function helps the robots to choose more informative actions. Fig. 4.5 shows the initial positions, the trajectories and the final positions of the robots. Unlike Fig. 4.4, robot 3, does not stay at the corner in Fig. 4.5. Mutual information makes the robots to move and gather more information, where robot 3 find a better action to sense a more worthy area.

In Fig. 4.6, the worth of the covered area by three robots are compared for our two experiments. Fig. 4.6 shows that without the mutual information term the robots sense 84% of the worth of the area at about step 250. Using mutual information the robots move frequently, gather more information, and converge to the Nash equilibrium faster. Here the robots can sense 84% of the worth of the area before the 100th step. This shows the faster convergence rate using the mutual information in our utility function.

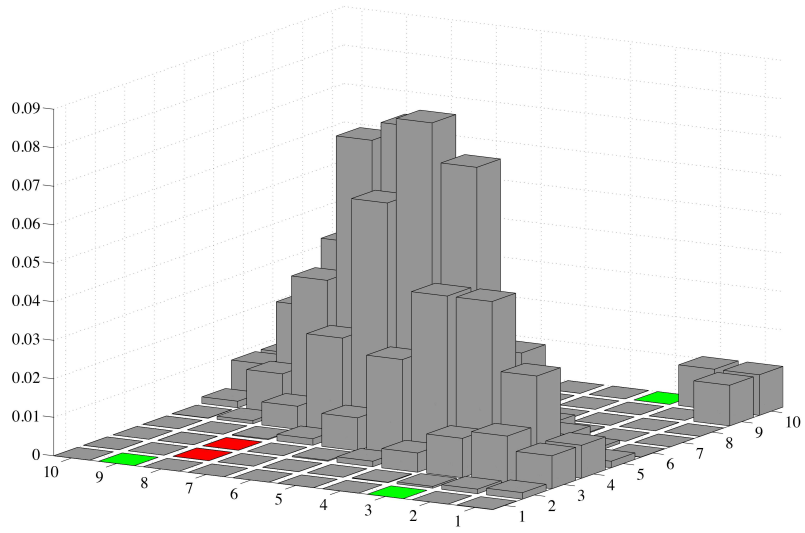


Figure 4.2: The worth of the area used for experimental validation. The initial positions of the robots and the obstacles are shown by green and red squares, respectively.

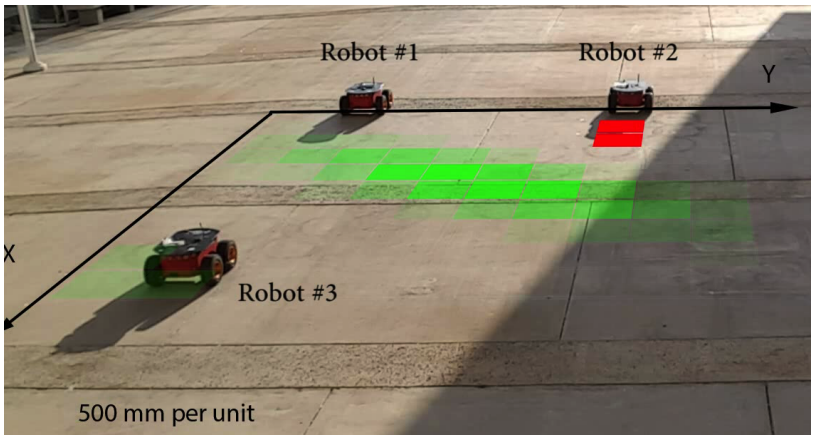


Figure 4.3: The initialization of physical experiment

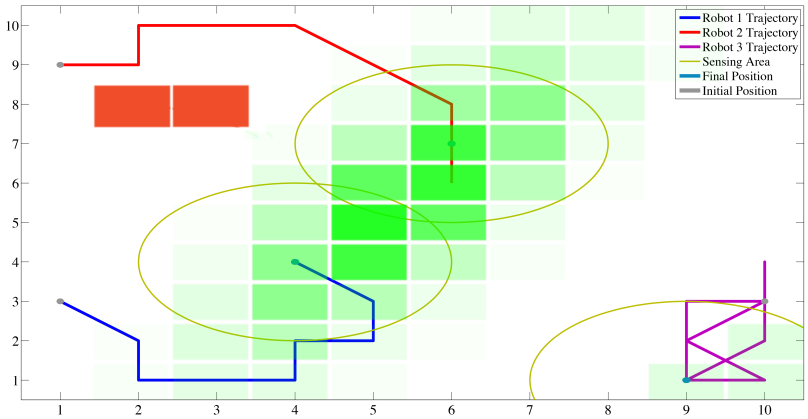


Figure 4.4: The trajectories and the final positions of the robots using the utility function (2.5) and for $\eta = 0$ (without mutual information).

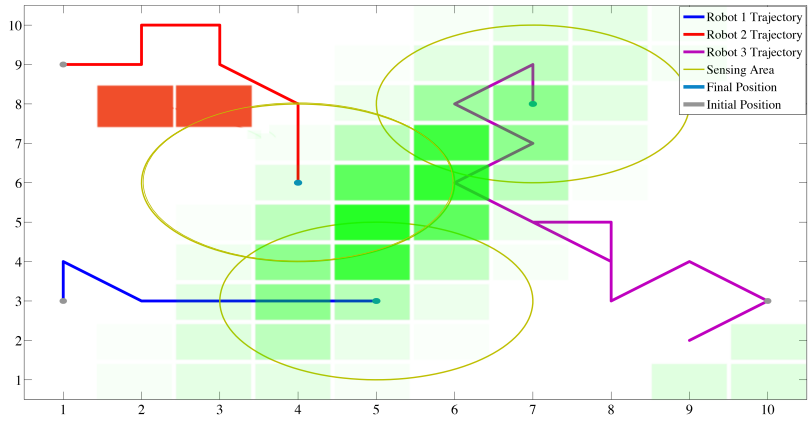


Figure 4.5: The trajectories and the final positions of the robots using the utility function (2.5) and for $\eta = \frac{0.01t+5}{t+10}$ (with mutual information).

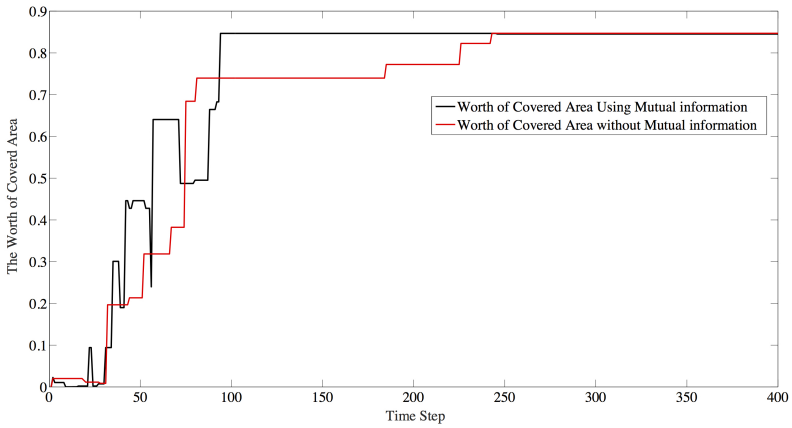


Figure 4.6: The worth of the covered area for two cases $\eta(t) = 0$ and $\eta = \frac{0.01t+5}{t+10}$.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis has two major parts. The first part is establishing a brief robot control system which can drive multiple robots to certain positions. As is shown in simulation and physical experiments, it works well.

The second part is implementing the game theory based algorithm on MobileSim and physical robots. The results demonstrate that this algorithm can drive robots to the configuration that covers most worthy area in an unknown environment eventually. For the case without mutual information, robot 3 has never left the corner which has relatively high worth. For the one with mutual information, robots move more often to collect more information as expected. Once robot 3 is ”forced” to leave, it will not go back because areas on the diagnoal have much more worth than the corner.

5.2 Future Work

There are many possible improvements that can be added to this thesis. For one, change the way to drive. Instead of changing linear and angular velocity periodically, we can make use of built-in classes such as ArActionGoTo. These classes are written by the developers of Pioneer. They may have better understanding of control on bottom layer which results in higher accuracy.

Another option is to add collision avoidence function because Pioneer is powerful and it may cause serious damage if they collide. Since the bottom layer control of Pioneer 3-AT will not change, a graphic user interface(GUI) may be developed to make it easier to control. Last but not least, we can get accurate amount of energy consumption by reading the values of robot's battery while in our experiment, we used estimated model instead.

References

[1] W. Ren, H. Chao, W. Bourgeous, N. Sorensen, and Y. Chen. Experimental validation of consensus algorithms for multivehicle cooperative control. *IEEE Transactions on Control Systems Technology*, 16(4):745–752, July 2008.

[2] Y. Cao, D. Stuart, W. Ren, and Z. Meng. Distributed containment control for multiple autonomous vehicles with double-integrator dynamics: Algorithms and experiments. *IEEE Transactions on Control Systems Technology*, 19(4):929–938, July 2011.

[3] Wei Ren and Nathan Sorensen. Distributed coordination architecture for multi-robot formation control. *Robot. Auton. Syst.*, 56(4):324–333, April 2008.

[4] J. Araujo, H. Sandberg, and K. H. Johansson. Experimental validation of a localization system based on a heterogeneous sensor network. In *Asian Control Conference, 2009. ASCC 2009. 7th*, pages 465–470, Aug 2009.

[5] A. Maali, L. Zeroukhi, A. D. Guerfi, G. Baudoin, and H. Mimoun. Experimental validation of the cooperative localization algorithm in wireless sensor network. In *Microwave Conference (EuMC), 2013 European*, pages 334–337, Oct 2013.

[6] S. Rahili and W. Ren. Game theory control solution for sensor coverage problem in unknown environment. In *53rd IEEE Conference on Decision and Control*, pages 1173–1178, Dec 2014.

[7] D. Pescaru, C. Istin, D. Curiac, and A. Doboli. Energy saving strategy for video-based wireless sensor networks under field coverage preservation. In *Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on*, volume 1, pages 289–294, May 2008.

[8] D. Fudenberg and D.K.Levine. *The Theory of Learning in Games*, volume 1. The MIT Press, January 1998.

[9] H.P. Young. *Strategic Learning and Its Limits*. Arne Ryde memorial lectures. Oxford University Press, 2004.

[10] J. R. Marden and J. S. Shamma. Revisiting log-linear learning: Asynchrony, completeness and payoff-based implementation. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 1171–1172, Sept 2010.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[12] Valliappa Lakshmanan and John S. Kain. A gaussian mixture model approach to forecast verification. *Weather and Forecasting*, 25(3):908–920, 2010.

[13] Y. Lim and J. S. Shamma. Robustness of stochastic stability in game theoretic learning. In *2013 American Control Conference*, pages 6145–6150, June 2013.

[14] M. C. Shewry and H. P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2):165–170, 1987.

[15] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn. Res.*, 9:235–284, June 2008.