Localization and Motion Coordination Among Multiple Collaborating Mobile Robots

A Dissertation Presented

by

Xionghui Lu

in Partial fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Mechanical Engineering

Stony Brook University

May 2011

Abstract

Localization and Motion Coordination Among Multiple Collaborating Mobile Robots

By

Xionghui Lu

Doctor of Philosophy

in

Mechanical Engineering

Stony Brook University

2011

This dissertation mainly addresses the problem of multi-robot motion coordination, including deployment algorithm and some other important issues in the deployment process, for example, localization and collision avoidance.

The first part of this dissertation is mainly about calibration and motion tests on the pioneer 3 DX mobile robot platform we are going to use. A pioneer 3 DX mobile robot has two driving wheels and one passive castor, and is a typical nonholonomic system. Due to nonholonomic constraint, the mobile robot has no nonzero side speed. In order to nicely control the robot and obtain its accurate running state, a motion calibration is necessary before serious experimental study. Three calibration parameters are adjusted manually until a well calibrated accuracy of motion is obtained. After calibration, a series of motion tests are conducted and the robot's capability of following commands is verified. Linear, rotational tests and a mixture of both movements are conducted in order, and the results show that the robot follows commands satisfactorily. The tests also provide some guides for our further experiments.

Though encoder reading from robot is very accurate once the robot is well calibrated, due to accumulative error in encoders, a global localization solution is desirable. In the second part, a localization algorithm based on single camera is proposed and followed by an error analysis. The algorithm is based on trilateration and pinhole camera model. Error analysis indicates that the algorithm has very high accuracy of localization.

In the next part, we discuss multi-robot deployment, which is a major of our work. A second order control method is proposed with our control frame work. This control framework belongs to the category of potential field methods. Compared with traditional deployment method that are based on position or velocity control, second order control is advantageous in making robots' movements smooth and natural. Besides, it is naturally incorporated in our control frame, which is originated from Hamilton's principle. By carefully designing the definition of artificial potential energy, the robots will move and be deployed automatically.

Another issue in robot deployment is collisions during the process. In order to address this issue, collision avoidance schemes are proposed. The first collision avoidance scheme firstly determines relative movement between two robots. If the distance is below the setting dangerous distance and moreover they are still moving closer, the collision avoidance scheme will be triggered. The method used in our collision avoidance is to increase resistant force acting on robots to slow them down immediately. The second collision avoidance scheme calculate each robot's position in the whole team based on local neighbors' information, and then spread out the robots layer by layer-outer robots move firstly and inner robot move once enough space is left out. This scheme aims to reduce collisions, eliminate unnecessary movements and save power.

The simulation results and experiments on Pioneer 3 DX mobile robots show that the deployment algorithm works well and the collision avoidance scheme can effectively reduce collisions during the deployment process.

The last part of our work is a collaboration work we have done with IBM, and the objective of the work is to provide moving power for their current Mobile Measurement Technology (MMT), which is used to scan and get thermal map of data centers for power management purpose. By using a PatrolBot mobile robot, a Bumblebee camera, a wireless router and a laptop, the upgraded system did a successfully scanning of IBM's Southbury data center with an operator controlling MMT remotely. The new system saves labor and time, and is more convenient to operate compared with the old MMT.

In the future, more work is on the way for the above several topics. Firstly, a sensor fusion method based on encoder and vision is desirable to obtain more accurate localization performance. Secondly, a more advanced and detailed study of collision avoidance is required to get a comprehensive understanding of collision avoidance. Thirdly, for MMT, a better user interface and more reliable localization method is also to be developed.

Table of Contents

LIST OF FIGURES VII
LIST OF TABLES X

CHAPTER 1 INTRODUCTION 1

1.1 MOTIVATION 1
1.2 RELATED WORKS 3
1.2.1 Localization Based on Trilateration 3
1.2.2 Multi Robot Deployment 5
1.3 OBJECTIVES 8
1.4 DISSERTATION STRUCTURE 9

CHAPTER 2 ROBOT MOTION CALIBRATION AND MOTION TESTS 11

2.1 INTRODUCTION OF PIONEER 3 DX MOBILE ROBOT 12
2.2 ROBOT CALIBRATION 13
2.3 LINEAR MOTION TEST 15
2.4 ROTATIONAL MOTION TEST 18
2.5 MIXED TEST INVOLVING BOTH LINEAR AND ANGULAR MOVEMENTS 20
2.6 DELAY OF RESPONSE IN BEGINNING OF MOVEMENTS 23
2.7 CONCLUSION 24

CHAPTER 3 SINGLE CAMERA LOCALIZATION ALGORITHM USING TRILATERATION AND ITS EXPERIMENTAL STUDY 26

3.1 LOCALIZATION ALGORITHM 27
3.1.1 Visual Angle Estimation 28
3.1.2 Position Estimation 29
3.1.3 Orientation Estimation 30
3.2 ERROR ANALYSIS 32
3.2.1 Camera Calibration Using Matlab Toolbox 34
3.2.2 Pictures Taken From Different Distances 35
3.2.3 True Positions and Orientations of Robot 37
3.2.4 Robot's Estimated Position and Orientation Using Our Localization Algorithm 40
3.3 CONCLUSION 48

CHAPTER 4 MULTI-ROBOT DEPLOYMENT, COLLISION AVOIDANCE AND EXPERIMENTAL STUDY ON PIONEER 3 DX 49

4.1 CONTROL FRAMEWORK 50
4.2 STUDY OF TWO KINDS OF POTENTIAL FIELD FORCES 53
4.2.1 Control Laws 53
4.2.2 Initial Robots Generation 55
4.2.3 Neighborhood Definition 56
4.2.4 Other Physical Constraints 57
4.2.5 Simulation Results 58
4.2.6 Real Experiment 63

4.2.7 Conclusion *66*
4.3 COLLISION AVOIDANCE SCHEME I – SLOWING DOWN 67
4.3.1 Results for Holonomic Robots *70*
4.3.2 Results for Nonholonomic Robots *77*
4.3.3 Real Experiment for Nonholonomic Robots *82*
4.4 COLLISION AVOIDANCE SCHEME II – LAYER BY LAYER SPREADING OUT + SLOWING DOWN 88
4.4.1 Results for Holonomic Robots *91*
4.4.2 Results for Nonholonomic Robots *94*
4.4.3 Energy Consumption Saving with Layer by Layer Spreading Out *97*
4.5 CONCLUSION 99

CHAPTER 5 ROBOTIZATION OF MMT 100

5.1 MOTIVATION AND OLD MMT 100
5.1.1 Requirements and Objectives of MMT *100*
5.1.2 Implementation *104*
5.2 ROBOTIC MMT 107
5.2.1 System Structure *109*
5.2.2 RMMT Platform *109*
5.2.3 Operator's Console *113*
5.3 HARDWARE COMPONENTS OF RMMT 116
5.3.1 Data Acquisition Triggering *118*
5.3.2 Encoder Pulses Mimicking *118*
5.4 RESULTS 120
5.5 CONCLUSION AND FUTURE WORK 123

CHAPTER 6 FUTURE WORK 125

BIBLIOGRAPHY 128

List of Figures

Figure 2.1: Pioneer 3 DX Mobile Robot....................12
Figure 2.2: Physical components of Pioneer 3 DX....................13
Figure 2.3: Real velocity VS setting velocity, Real acceleration VS calculated acceleration and accumulated average acceleration. 17
Figure 2.4: Setting acceleration VS accumulated average acceleration. 17
Figure 2.5: Setting rotational velocity and acceleration VS calculated velocity and acceleration. 19
Figure 2.6: Setting acceleration VS accumulated average acceleration. 19
Figure 2.7: Linear velocity and acceleration comparison....................20
Figure 2.8: Angular velocity and acceleration comparison 21
Figure 2.9: Trajectory comparison in mixed test....................22
Figure 2.10: Response delay at beginning of movements 24
Figure 3.1: Pinhole camera model 27
Figure 3.2: Check board pattern printed together with three circles....................32
Figure 3.3: Pictures used in camera calibration....................34
Figure 3.4: Radiuses of three circles....................36
Figure 3.5: Four photos of the pattern are taken from five different distances 36
Figure 3.6: Standard deviations 43
Figure 3.7: Standard deviations 47
Figure 4.1: Final deployment and average distances,....................60
Figure 4.2: Final deployment and average distances,....................61
Figure 4.3: Final deployment and average distances,....................62
Figure 4.4: Experiment 1 64
Figure 4.5: Experiment 2 65
Figure 4.6: Experiment 3 66
Figure 4.7: Collision detection....................69
Figure 4.8: Deployment simulation with control law (4.7) only 73
Figure 4.9: Deployment simulation with control law (4.8) only 74

Figure 4.10: Deployment simulation with control law (4.7) and (4.19) 75
Figure 4.11: Deployment simulation with control law (4.8) and (4.19) 76
Figure 4.12: Deployment simulation with control law (4.7) only 78
Figure 4.13: Deployment simulation with control law (4.8) only 79
Figure 4.14: Deployment simulation with control laws (4.7) and (4.19) 80
Figure 4.15: Deployment simulation with control laws (4.8) and (4.19) 81
Figure 4.16: Initial deployment of robots 83
Figure 4.17: After 2 minutes 84
Figure 4.18: After 4 minutes 84
Figure 4.19: After 6 minutes 85
Figure 4.20: Initial distribution 86
Figure 4.21: Distribution after 1000 steps 86
Figure 4.22: Distribution after 2000 steps 87
Figure 4.23: Final distribution, 3200 steps 87
Figure 4.24: Convergence of average distances 87
Figure 4.25: All four neighbors locate in the same semi-circle 89
Figure 4.26: Layer of Depth when maximum sector angle equals to PI 90
Figure 4.27: Layer of Depth when maximum sector angle equals to 1.2*PI 90
Figure 4.28: Deployment simulation with control law (4.7) and layer by layer spreading out strategy 92
Figure 4.29: Deployment simulation with control law (4.8) and layer by layer spreading out strategy 93
Figure 4.30: Deployment simulation with control law (4.7) and layer by layer spreading out + slowing down 95
Figure 4.31: Deployment simulation with control law (4.8) and layer by layer spreading out + slowing down 96
Figure 5.1: Typical equipment placement in data centers 101
Figure 5.2: IBM MMT sensing cart 104
Figure 5.3: Interface Box 105
Figure 5.4: Handler 105
Figure 5.5: Temperature sensor 106
Figure 5.6: Tmapr interface 106
Figure 5.7: New MMT under work 107
Figure 5.8: Functional diagram of the onboard RMMT software 111

Figure 5.9: Sensor cart localized in the data center 114
Figure 5.10: Feedback from onboard camera 115
Figure 5.11: Mechanical integration between MMT and robot......................................116
Figure 5.12: User Interface 117
Figure 5.13: NI USB 6501 118
Figure 5.14: Encoder pulses..119
Figure 5.15: Images from environment-attached cameras..121
Figure 5.16: Flowchart of the data center scanning process 122
Figure 5.17: Maps of temperatures at 0.5 (upper) and 4.5 (lower) feet above the ground123

List of Tables

Table 3.1: Real relative position and orientation of robot when 2 meters away 37

Table 3.2: Real relative position and orientation of robot when 2.5 meters away 38

Table 3.3: Real relative position and orientation of robot when 3 meters away 38

Table 3.4: Real relative position and orientation of robot when 3.5 meters away 39

Table 3.5: Real relative position and orientation of robot when 4 meters away 39

Table 3.6: 2 meters way 40

Table 3.7: 2.5 meters way 40

Table 3.8: 3 meters way 41

Table 3.9: 3.5 meters way 41

Table 3.10: 4 meters way 42

Table 3.11: Summary of standard deviations 42

Table 3.12: 2 meters away 44

Table 3.13: 2.5 meters away 44

Table 3.14: 3 meters away 45

Table 3.15: 3.5 meters away 45

Table 3.16: 4 meters away 46

Table 3.17: Summary of standard deviations 46

Table 4.1: Number of collisions when n=1..60

Table 4.2: Number of collisions when n=2..61

Table 4.3: Number of collisions when n=3..62

Table 4.4: Rate of convergence 76

Table 4.5: Rate of convergence 81

Table 4.6: Rate of convergence 93

Table 4.7: Rate of convergence 96

Table 4.8: Energy consumption comparison 98

Table 5.1: Movement meaning of encoder pulses 120

Chapter 1 Introduction

Robots have been employed more and more widely in various applications, for example, security surveillance, environment monitoring, space exploration, disaster rescue, and etc. They are replacing human beings doing boring, dirty and dangerous work.

Traditionally, industrial applications focused more on single robot. In contrast, multi-robot system is a relative new topic. However, due to multi robot system's superior capabilities over single robot, it has shown greater and greater potential in a lot of applications.

In this dissertation, we proposed a decentralized deployment algorithm for multi robot system, introduced a global localization method based on single camera and two collision avoidance schemes to eliminate collisions during deployment process.

1.1 Motivation

Robots have been taking over many work from human being in the past a few decades, this trend will keep going in future.

As a matter of fact, traditional applications focused more on single robot. In contrast, multi-robot system is a relative new topic. However, multi robot system shows superior capabilities over single robot, for example, multi-robot system has wider environment probe capability, and it is more robust to external interruptions and internal failures, more importantly, with good coordination, multi-robot system is able to get job done more efficiently. As a result, it has shown greater and greater potential in a lot of applications, such as planetary exploration, disaster rescue, environment surveillance, etc.

In study of multi-robot system, multi-robot deployment is a fundamental problem. It is the starting point for more advance applications of multi robot system and not surprisingly has attracted a lot of industrial and academic attention. Our research aims to formulate a distributed method for multiple robots deployment and verify it using our Pioneer 3 DX robot platform.

The first unique of our work is that it determines a robot's motion merely based on limited information acquired from self-equipped devices and local neighbors, while centralized control method requires global information, which is not accessible in many real applications. Due to this characteristic, distributed systems rely more on collaboration between all members rather than putting all eggs in the same basketball and all commands come from a single leader. Equally importantly, it requires lower computation ability for each member, making system more cost-effective. The other point of our job is, compared with most traditional multi robot deployment algorithms based on position and velocity control, our method goes directly to second order control – acceleration control. Acceleration control makes implementation of algorithms in specific hardware platforms simple and straightforward, and it makes movement smooth and natural. In a word, decentralized second order control in multi robot deployment summarizes our major work.

During deployment process, two issues need to be addressed: localization and collision avoidance. Usually, given initial positions and orientations, self-equipped encoders are able to calculate a robot's positions and orientations based on history movements. However, accumulative errors limit encoders' usability in demanding applications and a global localization method is desirable. In our work, using a single camera, a localization algorithm based on trilateration is proposed to provide a more accurate global localization solution. The second issue, collision avoidance, is solved by two schemes we proposed. These two schemes fit our control frame in that they use local information and eliminate collisions via second order control.

Another job of us is to upgrade IBM's old MMT (Mobile Measurement Technology) and power it with a mobile robot. Their current MMT is used to monitor temperature and moisture in data centers. It doesn't have moving capability and completely rely on human being. Our objective is to integrate their current measuring ability with mobile robot's moving capability, therefore in the future operator can remotely control the new MMT and eventually it can move, measure and upload data totally on itself. This would save a lot of labor and reduce cost significantly.

1.2 Related Works

1.2.1 Localization Based on Trilateration

The principle behind trilateration is to use distances from at least three different landmarks to determine object's current position and orientation. This method, essentially, is equivalent to finding intersection points between three or more spheres. Manolakis[1][2] proposed closed-form solution to trilateration problems. Coope[3] further proposed two ways of solving trilateration problems, one is based on Gaussian elimination and the other applies orthogonal decomposition and transformation. A nonlinear least-square optimization method for obtaining approximate results is proposed in this chapter. Thomas and Ros[4] introduced a general formulation for the closed-form solution based on Cayley-Menger determinant defined by constructive geometric argument. Fang[5] proved that trilateration problem could be simplified by defining the problem according to a frame.

Many applications have been developed based on the principle of trilateration. The Global Positioning System (GPS) uses trilateration to locate a receiver based on the travel distances of radio frequency signals from the satellites [6]. It provides a powerful tool for outdoor mobile robot localization and navigation [7]. In indoor environments, some localization systems have been developed, such as SpotOn [8], Active Badge [9], Active bat [9], Cricket [11], based on radio frequency [7], infrared [8], and ultrasound [9, 10] signals respectively. Recently, Zhou et al. have also introduced a new indoor localization method for mobile robots based on the laser-activated RFID landmarks [12].

Triangulation, instead of using distance measurements, uses bearing measurements among references to locate an object. The basic idea of triangulation is that in a plane containing the references and object, the object is located at the intersection of the circles each of which is determined by two references and the bearing between them. Cohen compared four solution methods for triangulation, i.e. iterative search, geometric triangulation, geometric circle intersection and Newton-Raphson iterative method, and showed that geometric circle intersection is the most robust one among them [13]. Betke and Gurvits presented a position estimation algorithm using the complex numbers representation of the landmarks [14]. The main advantage of this algorithm is the linear time complexity with respect to the number of landmarks and the robustness to the noisy input. Shimshoni also presented an algebraic solution by applying several transformations to the linear system of equations which are defined by triangulation constraints [15]. He showed that these transformations indeed improved accuracy. Sutherland and Thompson discovered that the position error is influenced by both the input bearing error and the distribution of landmarks [16].

Vision sensors are widely used to measure the bearings. The structured features, such as doors and wall corners, are extracted from images. The 2D bearings of landmarks can be recovered from vertical edges. Muñoz and Gonzalez developed a 2D landmark-based triangulation algorithm in which the bearings are derived from a single image [17]. Other systems utilizing the bearing measurements for localization can be found in [14, 18-20].

1.2.2 Multi Robot Deployment

Due to advantages of second order control, it becomes basis of our deployment algorithm. The deployment algorithm works seamlessly with the collision avoidance under a global control frame we proposed, which is based on Hamilton's principle. This principle applies to not only classic mechanics but also classic fields, like electromagnetic and gravitational fields. In our case, this principle states that dynamics of robots could be determined by a variational problem for a function based on its Lagrangian, which contains all physical information of the robot and the forces acting on it. By carefully defining artificial energy for the robot, which is considered part of the Lagrangian, the robots would move automatically to meet our deployment requirement.

This second order and natural control algorithm also owns another advantage in that it is a distributed control law. Many traditional multiple robots deployment methods depend on a central command center, which collects data from robots and sends commands back after a calculation based on complex algorithms. This command center could be either one of the robots or a separate computing unit [21, 22]. Though effective in a stable environment, due to its high communication complexity, it is slow in response to abrupt changes in the system and environment. Moreover, it demands high processing power for the leader, and is vulnerable to failure of the leader. In contrast, decentralized control allows individual robots to behave according to local information and achieves global goals by following distributed control laws, which can be highly adaptive to environment changes and individual failure. Recognizing its advantages, the research on multi-robot deployment has been largely leaning towards decentralized control.

For the concept of artificial potential/force field, it has been widely adopted in decentralized deployment control. For multi-target observation, Parker defined a distributed control law in terms of force fields attractive for nearby targets and repulsive for nearby robots, weighted by the probability of target existence and not being observed [22,23]. For large system deployment, Reif and Wang proposed a "social potential field" method to reflect social behaviors such as clustering and escorting [24]. To deploy a mobile sensor network in an unknown environment from a compact initial configuration, Howard et al. defined a potential field in which each node is repelled by both obstacles and other nodes [25]. To maximize the sensor coverage while maintaining the number of connected neighbors, Poduri and Suktame used the repulsive forces among nodes to improve their coverage and the attractive forces to prevent the nodes from losing connectivity [26]. Popa et al. also defined an attractive force to maintain inter-node connections [27]. Ji and Egerstedt presented an approach based on weighted graph Laplacians and the edge-tension function to control multi-agent rendezvous and formation while maintaining communication connections [28]. Moreover, for deploying mobile sensor networks to approach an isometric grid, Lam and Liu defined a force field based on the difference between current and ideal local configurations [29]. To cover a moving target, Jenkin and Dudek formulated the problem as a global energy minimization task over the entire collective in which each robot moves in the gradient descent direction of its local estimate of the global energy [30]. In addition, event-driven schemes were presented by Butler and Rus to deploy mobile sensors toward the distribution of the sensed events [31]. With the intention to reduce the communication complexity, Tan defined the potential field for each robot based on only its one-hop neighbors of the Delaunay triangulation [32].

Besides potential/force field methods, Cortes et al. showed that an optimal coverage of multiple mobile sensors is provided by the centroidal Voronoi partition where each sensor is located at the centroid of its Voronoi cell, and presented a gradient decent algorithm to move sensors towards the centroidal Voronoi configurations [33]. A similar method was adopted in [34], and a slightly different method based on the r-limited Voronoi partition was presented in [35]. Schwager et al. proposed an adaptive decentralized controller to drive a team of networked robots to the estimated centroids of their Voronoi regions while improving sensory distribution over time [36].

Moreover, diffusion-based schemes were also proposed for dynamic coverage of bounded environments, including random diffusion [37], gas diffusion [38], and fluid diffusion [39]. In addition, Jung and Sukhatme presented a region-based method in which the robots are deployed according to a compromise between robot densities and target densities in all the regions with a consideration of robot travel distances, and robots are positioned inside a region optimally to cover all the targets [40].

Although decentralized multi-robot deployment control has received a lot of attention, existing methods mostly focus on general schemes. Very limited consideration has been given to practical issues arising in realistic multi-robot deployment processes, such as the constraints in kinematics, dynamics and communications and the problem of collision avoidance. For instance, the nonholonomic constraint arising in wheeled robots has been largely ignored by assuming holonomic drive [25] or using the simple unicycle model [33,34]. This decoupling between the high-level control law and the implementation layer may cause unexpected results or even failures in realistic multi-robot deployment processes.

Targeting the robust implementation in realistic multi-robot systems, in a recent work [41], we proposed a decentralized deployment control framework, which integrates the concept of artificial potential field with Hamilton's principle [42] of the classic mechanics to generate the control law for robot self-deployment motion. It naturally incorporates the nonholonomic constraints arising in wheeled robots. Built on our previous work, this chapter will present a control law for reliably establishing desired sensor coverage and maintain communication connections in an environment from a compact initial multi-robot gathering, and particularly address the issue of collision avoidance during the deployment process. This system depends heavily on this central commander, and is very vulnerable to environment interruptions. Once the central commander fails, the whole system is unable to work at all. Besides, since the robots' movements totally depend on commands from central unit, it would don't know their next step once they lose communications.

1.3 Objectives

This dissertation focuses on proposing a decentralized multi-robot deployment algorithm based on second order control and addressing collisions and localization issue in the deployment process. Our work is summarized as following:

1) Calibrate robots and test their motion capability;
2) Propose a global localization algorithm based on single camera;
3) Propose a distributed multiple mobile robot deployment algorithm exclusively for nonholonomic robots;
4) Address collisions among robots during multi robots deployment;

5) Collaborate with IBM and upgrade their current data center monitoring system to be mobile robot powered.

1.4 Dissertation Structure

Chapter 2 makes a brief introduction of the robots we are going to use and describes steps of motion calibration, which guarantees nice control of robots and accurate feedback of robots' running state, like velocity. Besides, a series of motion tests are done to verify motion capability of the robot, to evaluate how well the velocity and acceleration read from encoders follow our settings. For a well calibrated robot, velocity read from encoders are very close to robot's actual velocity, so to some extent that the tests reflect how accurately the robots execute commands. The results would give us a good guide for further serious experiments.

In chapter 3, a localization method based on single camera is introduced and followed by an experiment to analyze its accuracy. The method depends on single camera feedback and use trilateration in calculation, which is the same mechanism used in GPS.

Chapter 4 proposes a distributed algorithm for multiple robots deployment and further discusses collision avoidance in deployment process. The decentralized algorithm falls into category of potential field method and conducts second order control, which leads to natural and smooth movements of robots. For collision avoidance in deployment, two schemes are proposed.

In Chapter 5, by cooperating with IBM Watson Research Center, a PartolBot mobile robot is integrated with their mobile measurement system, which is used to monitor temperature and other thermal data of data centers. Instead of depending on being pushed around the big data centers by human operators, the new system is able to be operated remotely and collects data in a more convenient way.

Chapter 6 gives the conclusions based on work we have done and list the work we are going to do in the future.

Chapter 2 Robot Motion Calibration and Motion Tests

It has a long way to go from computer simulations to real experiments because real experiments have a lot of uncertainties that simulations are unable to forecast or follow. For example, in simulations, distance that the robot moves in a certain period of time is exactly equal to velocity multiplied by time interval, which is not the case in real experiment. Besides, delay of command execution is not taken into account in simulations. In fact, in real experiments, robots can’t even maintain absolute constant velocities, and slow response time of wheel motors sometimes undermines algorithm performance significantly. A lot of experiments have illustrated that even given the same initial settings, robots’ movements were still slightly different in the same environment. In a word, real experiment involves numerous hardware implementation details, which result in noises and uncertainties. Therefore, simulations can never replace real experiment and a successful algorithm should bear tests with real hardware in real environment, and this is an inevitable step to serious applications in real life.

In our work, Pioneer 3 DX mobile robots (Figure 2.1) are chosen to be our experimental platform, which are commercially available from MobileRobots Inc.. This chapter is mainly about their motion capability test.

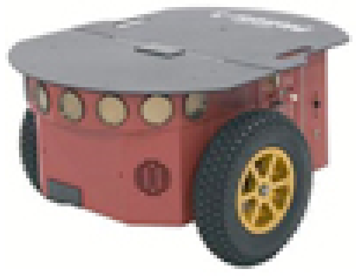


Figure 2.1: Pioneer 3 DX Mobile Robot

This chapter is organized as followed. Firstly, in order to better control robot and obtain accurate motion state of robot, the robot needs to be calibrated. After that, a series of motion tests are conducted to verify its moving ability and accuracy. A linear, rotational movement test and mixture of both are conducted. As a result, in the end of this chapter, we illustrated that calibrated robots run accurately and follows commands well. The tests also provide some programming guides for our later experiments.

2.1 Introduction of Pioneer 3 DX Mobile Robot

Pioneer 3 DX weights 9 kg (20 pounds with one battery) with height of 21.5 cm and width of 38cm [43], and it has two differential driving wheels and one passive castor. In front and rear body the robots are equipped with 8 sonar sensors respectively, (see Figure 2.2). Besides, the robot is upgradable for more powerful capabilities. For example, equipment of arms can help manipulate things, laser scanners are able to detect environment layout, camera's feedback can be used to observe environment and helps robot make smarter moving decisions.

Just like most of other commercially available robots, the robot uses client-server working mode. The microcontroller handles all low-level details and converts all high level commands to low level executable commands, like driving wheels rotate and reading data from every sensor, encoders and sonar sensors, to name a few. An onboard computer is attached with Pioneer 3 DX mobile robot via RS 232. Program running on PC communicates with the robot constantly and tells it its next action.

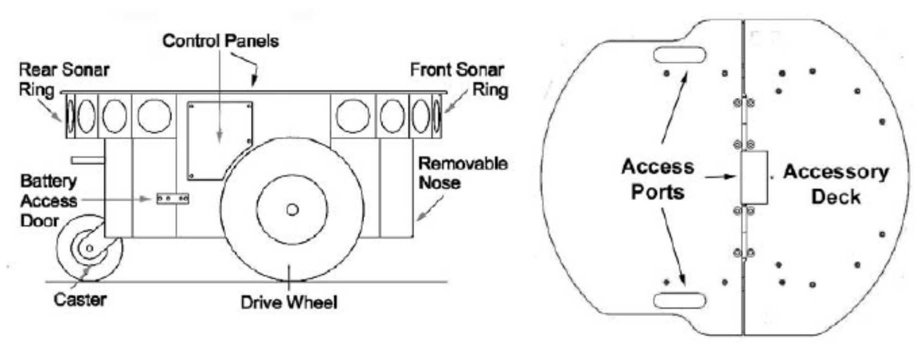


Figure 2.2: Physical components of Pioneer 3 DX

The bridge between high level and low level control is ARIA, which stands for Advanced Robotics Interface for Application. It is a C++ based open source development library that insulates users from low level details and helps users to focus on high level algorithm development and verification. With hundreds of well designed functions, users could give simple commands directly to robot and read back usable data from sensors. It makes controlling of robot much easier and reliable. Moreover, ARIA also reserves necessary interfaces for potential extensional devices, like arms and cameras.

2.2 Robot Calibration

In order to nicely control robot and obtain robot's running state, a robot calibration is needed before being used in real experiments and applications. A robot needs a new calibration once it is moved to a new environment, or the old environment has just experienced any kind of change. For example, the floor is just polished, or an air conditioner is just replaced. Changes of environment lead to changes of some physical parameters, including tire pressure of robots, which results in small change of tire's diameter and friction coefficient with ground. Besides, for the same reason, regular inspection of robot's tire pressure is necessary. However, calibration doesn't guarantee absolute accuracy of robots' motion, instead, it manages to improve robots' performance as high as possible.

Fortunately the calibration is not very hard for Pioneer 3 DX. Robot's moving accuracy could be adjusted by three parameters stored in microcontroller's flash memory. These parameters are DriftFactor, RevCount and Ticksmm.

The first one, DriftFactor, controls robot's drift in linear movements. Robots usually drift to right or left side after travelling a certain distance. DriftFactor controls moving consistency between two wheels and makes sure robot wouldn't drift to either side. Since DriftFactor affects both RevCount and Ticksmm, it needs to be adjusted at first place.

The way of adjusting DriftFactor is simple. We use a demo program to make robot move forward 1 meter straightly and observe which side the robot has drifted to. If it drifts to right side, reduce value of DriftFactor and vice versa. The same principle of adjustment applies to RevCount and Ticksmm.

RevCount is the differential number of encoder ticks for a 180-degree rotation, and apparently it is related to accuracy of rotational movement. Similarly, Ticksmm is number of encoder ticks for every millimeter the wheels rotate, and it is related to accuracy of translational movement.

The calibration of RevCount and Ticksmm is very similar to that of DriftFactor . A demo program is used to command robot move 1 meter forward and then its actual moving distance is measured. If the actual distance is less than meter, then we should decrease value of Ticksmm, otherwise, we increase it. Eventually, a proper Ticksmm is obtained. RevCount can be calibrated either before or after calibration of Ticksmm. With the demo program, and given initial direction of zero, robot is asked to rotate 180 degrees, and its actual rotation angle tells me to increase or decrease Ticksmm.

After the calibration, our records indicate that linear motion error could be limited to less than 1cm per 3 meters, rotational error is less than 0.9 degree per 180 degree. To sum, with proper choose of the three values, the robot moves with very good accuracy, more importantly, velocity reading from encoder (via ARIA) is trustable and usable in our later experiments.

2.3 Linear Motion Test

After robot calibration is done, a series of motion tests are conducted to further verify moving performance of robots. These tests are mainly used to evaluate robot's moving capability and accuracy. In this section, linear motion tests try to verify robot's capability of moving forward in high accuracy without drifting to either side,

To do these tests, three programs are written. In the program, robot is commanded to do 5 movements in sequence, each movement lasts 4 seconds. Firstly, robot accelerates from speed of 0 to 40 mm/s in 4s (means acceleration is 10 mm/s^2), and then decelerates back to zero (at -10 mm/s^2), then stays stationary for 8s before repeating the acceleration and deceleration process one more time.

During the whole test, real velocities and setting velocities are recorded for later comparison. The real velocities of robot are based on constant reading using function getVel()

from ARIA (for a well calibrated robot, its actual velocity is very close to velocity that is read from encoders, and we assume that it is the actual velocity).

Actual accelerations are calculated by deriving actual velocity over time. In our case, dividing velocity change by according time period leads to average acceleration during this small slice of time. These accelerations are also compared with setting accelerations.

From Figure 2.3 and unsurprisingly, setting velocities and acceleration are very smooth since they are set manually, and actual velocities and accelerations fluctuate seriously mainly due to noises and response delay.

In order to compare actual acceleration with setting acceleration in a more reasonable and clear way, a third variable named accumulated average acceleration is taken into account, which is calculated in this way.

$$a_r = \frac{v_c - v_s}{t_c - t_s} \tag{2.1}$$

v_c is current actual velocity, v_s is the actual velocity at beginning of this period, t_c and t_s record current moment and the beginning time of this period. A locally amplified image of accumulated acceleration and setting acceleration is displayed in Figure 2.4. As we can see that the accumulated acceleration approaches the setting acceleration closer and closer along with time.

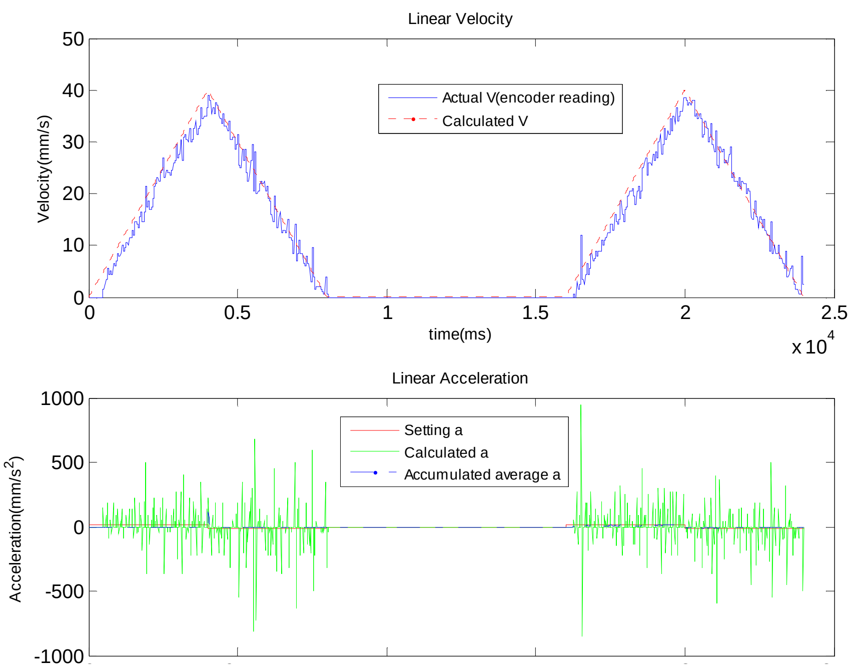


Figure 2.3: Real velocity VS setting velocity, Real acceleration VS calculated acceleration and accumulated average acceleration.

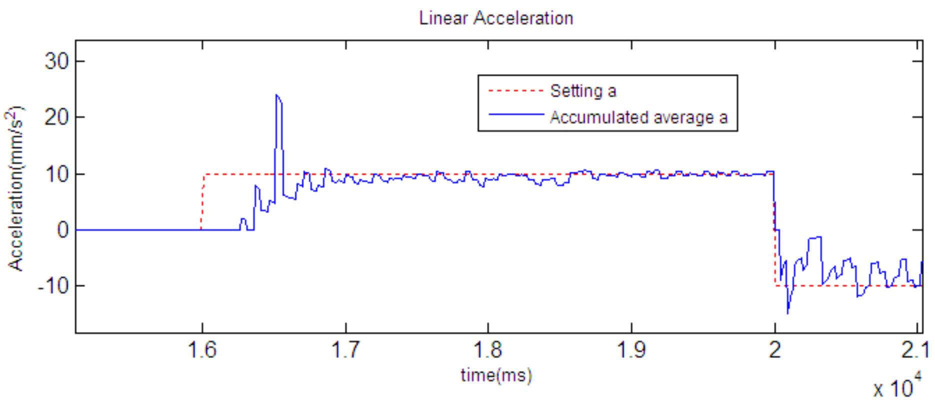


Figure 2.4: Setting acceleration VS accumulated average acceleration.

2.4 Rotational Motion Test

The second test mainly focuses on robot’s rotational performance. It is to verify how well the robot’s real rotation follows the command. The whole process consists of 7 stages and basically the same as the linear test. Each stage lasts 4 seconds. During the first period, robot rotates and accelerates to 40 deg/s (CCW in top-down view) in 4s, and decelerates from 40 deg/s to -40 deg/s (CW) in 4s, then accelerates again to a larger velocity of 80 deg/s(CCW) in 4s and slows down to 40 deg/s(CCW), then speeds to 80 deg/s and back to static, then decelerates to -40 deg/s. The whole process lasts 28 seconds.

Figure 2.5 shows how real rotational velocity follows setting velocity. As what we did in linear test, accumulated average angular acceleration is calculated and compared with setting acceleration. The results are shown in Figure 2.6. The results are not surprising as they are consistent with results we got in last section. The actual rotation doesn’t follow the commands strictly, but on average, the two are matched in a good level. At beginning of the test, there is a lag between them and this phenomenon will be discussed later.

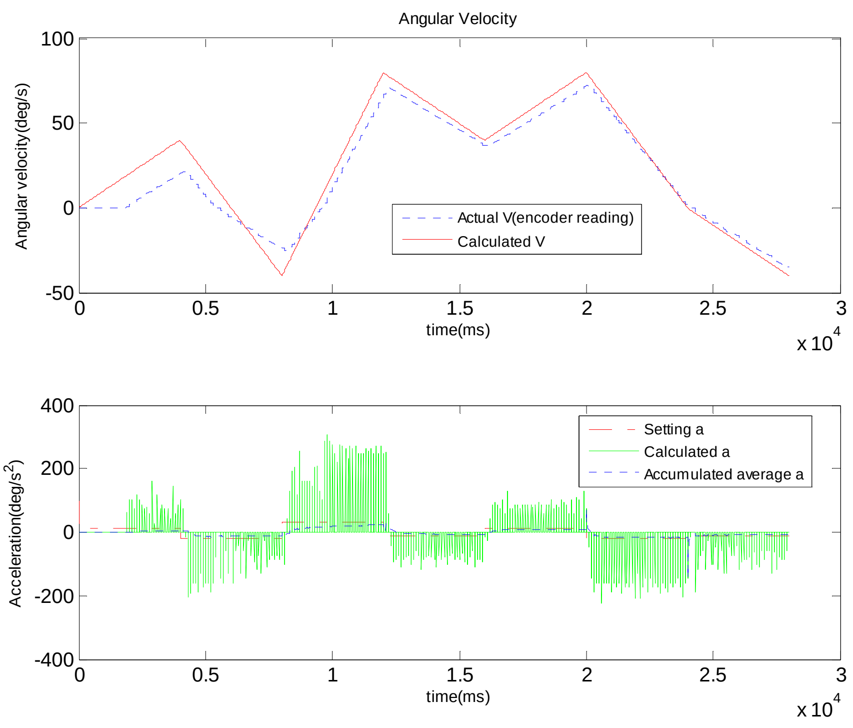


Figure 2.5: Setting rotational velocity and acceleration VS calculated velocity and acceleration.

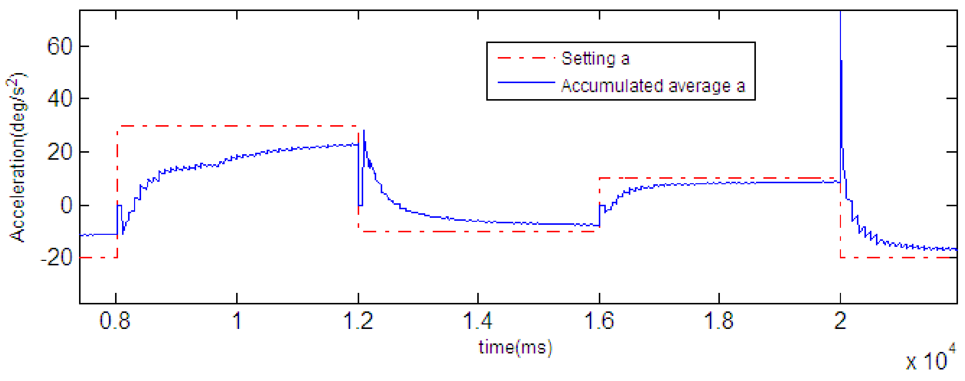


Figure 2.6: Setting acceleration VS accumulated average acceleration.

2.5 Mixed Test Involving both Linear and Angular Movements

The third test takes linear and angular movements together. The test consists of 6 stages. As we can see from Figure 2.7 and Figure 2.8 that the robot firstly moves forward and reach velocity of 80 mm/s in 4s and then slows down to static in the following 4s. Then, it keeps position and accelerates to rotational velocity of 40 deg/s and slows down to static. After this, another round of translational movement is repeated.

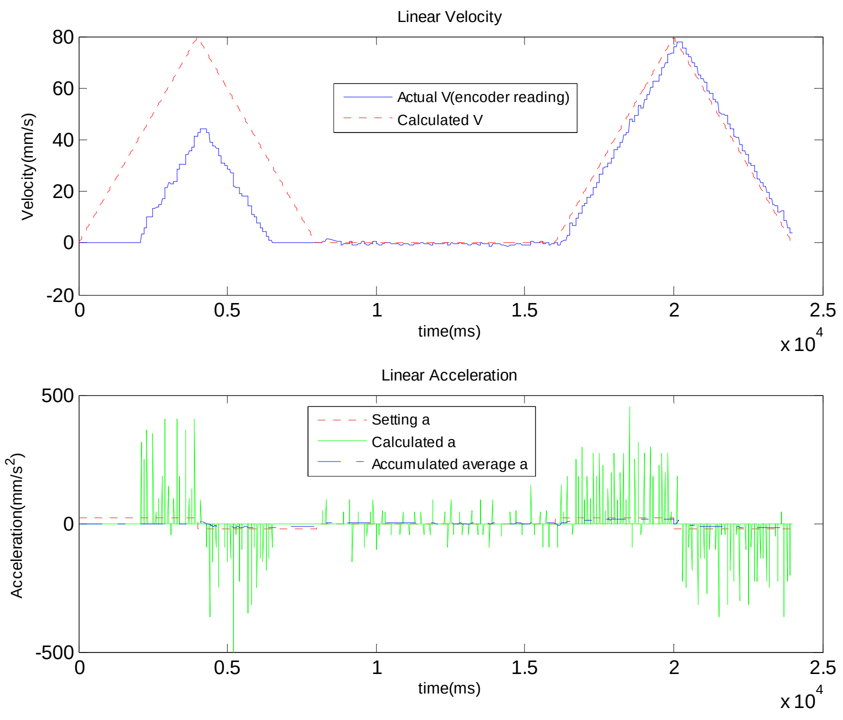


Figure 2.7: Linear velocity and acceleration comparison

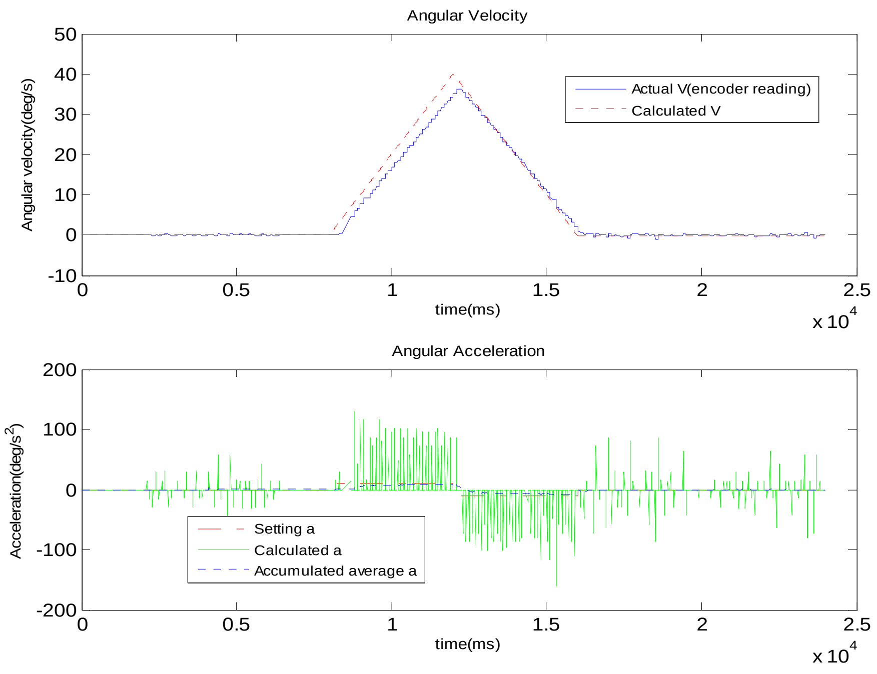


Figure 2.8: Angular velocity and acceleration comparison

Figure 2.9 displays two trajectories of the robot, one trajectory is the actual trajectory, which based on reading via ARIA function (we actually measured the real trajectory and it is almost the same as the trajectory read from encoders). The other is trajectory calculated based on real time velocity integration over time. Distance travelled at every command cycle is:

$$d_c = \frac{(v_s + v_e)t_{cycle}}{2} \tag{2.2}$$

where v_s and v_e are actual velocities of robot at beginning and end of the running cycle, t_{cycle} represents time interval of the cycle.

Figure 2.9 shows that real trajectory is very different from ideal trajectory, which we believe is caused by accumulated velocity error due to response delay and noises. Actually, from Pioneer 3 DX's design philosophy, the movement of robot is more based on position and direction control, and robot keep adjusting moving velocities based on encoder reading feedback. From our other tests, the robot is much better at positioning control, which is based on continuous reading from encoder and makes robots' movement very accurate. To sum up, if we control robot by controlling its velocity or acceleration, position and direction information calculated from integration of velocity is not reliable, and reading from encoders is more accurate and trustable.

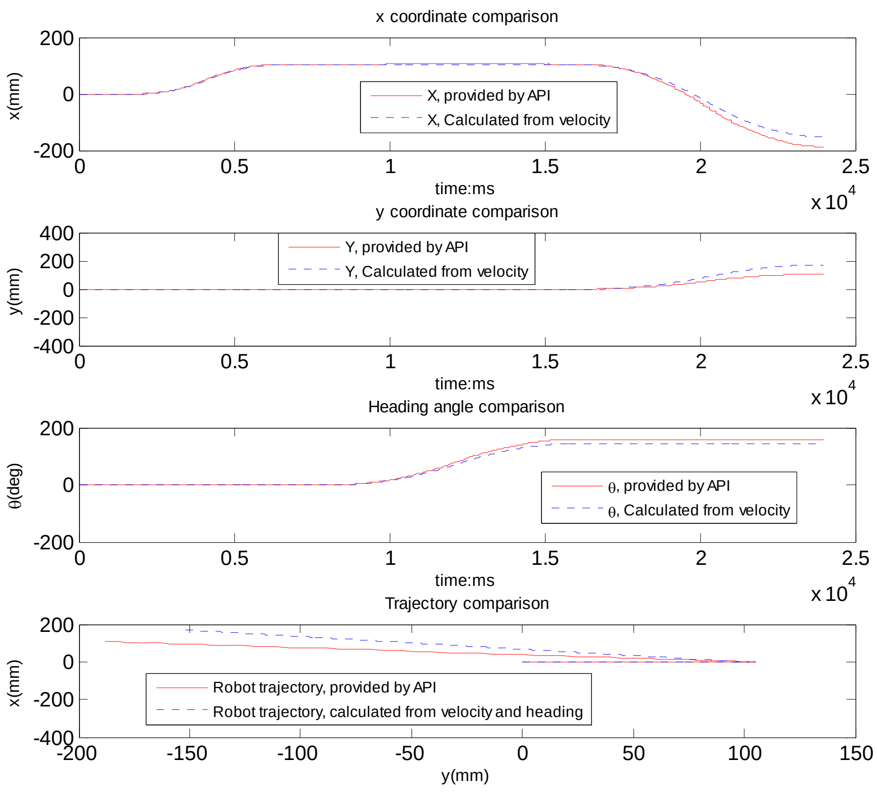


Figure 2.9: Trajectory comparison in mixed test

2.6 Delay of Response in Beginning of Movements

As we have noticed earlier from the three tests that there was always a delay of response in the beginning of movement. In Figure 2.3, Figure 2.5 and Figure 2.7, we can clearly see the delays, whose lasting time varies every time. In the first test, it lasts around 480ms, in the second test, it lasts around 1.8s, and 2.1s for the third tests. The delays are amplified and displayed in Figure 2.10. The delays, in our mind, are due to a variety of factors. Firstly, the initializations of programs, including high level program we wrote and the low level firmware, make part of the delays. Secondly, initialization of hardware also takes time. Thirdly, the design philosophy of Pioneer 3 DX robots doesn't promise to give extremely high accurate velocity output, instead, its philosophy puts high accuracy of position and orientation displacement at its highest priority and this is understandable since so far most robot experiments are based on position and angle control.

The experiments illustrate the existence of the delay at beginning of experiments and the solution, as we found out, is to give an empty standing command. The command leaves some time for the robots to release the delays.

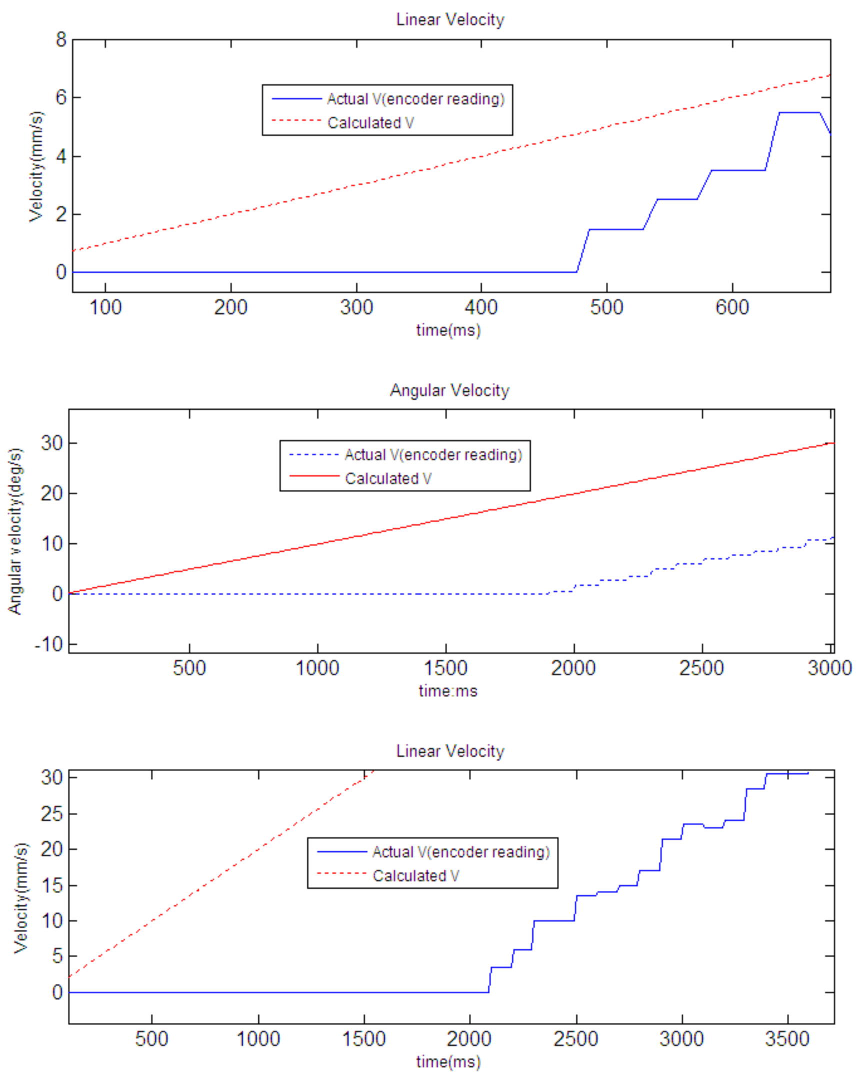


Figure 2.10: Response delay at beginning of movements

2.7 Conclusion

In order to obtain a robot with accurate moving capability, a Pioneer 3 DX mobile robot is carefully calibrated and the calibrated robot achieves very good level of moving accuracy thereafter. The position and orientation of a calibrated robot based on encoder reading is trustable in later experiments that last short period of time.

After the calibration, a sequence of three tests is conducted to verify robot's linear and rotational moving capability and accuracy. The results show that robot's actual velocity and acceleration follow setting parameters satisfactorily.

Besides, delay of response exists and varies from time to time and the solution is to give an empty motion command at beginning of movements. Partial cause of the delays is that commands take time to reach low level microcontroller and wheel motors based on PID control always have a delay of response.

The work we have done in this chapter provides some guides for our further experiments. For application of algorithm in Pioneer 3 DX, our second order control calculates desired accelerations other than velocities or positions. However, in Pioneer 3's design philosophy, setting acceleration doesn't make robot move, it is just a value stored in microcontroller's flash memory to control how fast velocity increases. In order to make robot follow desired acceleration, command velocity (which is either bigger or less than current velocity) need to be set and sent to motor controller continuously. This applies to both angular and linear acceleration control.

Finally, accuracy of position and orientation of a well calibrated robot doesn't last very long time due to encoders' accumulative error. To solve this problem a global localization method is desirable. Actually, a method based on single camera will be introduced and discussed in next chapter.

Chapter 3 Single Camera Localization Algorithm Using Trilateration and Its Experimental Study

Localization is a fundamental problem in robotics, especially in robot navigation field. It takes advantages of various kinds of sensory devices and algorithms to calculate and improve accuracy of robot's current position and orientations. A lot of work has been done in this field, and generally they can be divided into several categories. In the first category, encoders of wheels could be used to calculate distance offset that each wheel has travelled and calculate robot's current position and orientation from robot's initial positions and orientations. However, this method can't avoid accumulative error and can't be used in position accuracy demanding areas. The second one is use artificial landmarks for references, for example, for mobile robots, some magnetic lanes is embedded in the floor and robot can detect magnetic field and thus determine its current position. Recently, methods based on camera feedback are attracting more and more people's attention due to its enriched information of environment. Our lab proposed a method based on trilateration that calculate robot's position and orientation based on single image of identified landmarks which is equipped on top of the robot.

The main benefits of using single camera over stereo vision system are as following. Firstly, from cost perspective, stereo vision system doubles the budget compared with single camera. It is a lot of money if dozens of robots to be equipped. Secondly, for location based on landmarks, stereo vision system always requires landmark matching before calculating localizations. Thirdly, stereo vision system usually requires more computing power since it has much more image processing compared with algorithm based on single camera.

This chapter is organized as followed. In the first part, the trilateration algorithm is described in details. After that, an experiment is described and conducted under different circumstances to verify accuracy of the algorithm. Experimental results are listed in tables and drawn in figures. The conclusions are on the last section.

3.1 Localization Algorithm

The localization algorithm is to be introduced in this section. The visual angle between two landmarks can be calculated from their projections in the same image. By using perspective geography, the constraints could be written into a group of equations, distances from optical center to landmarks could be calculated thereafter and eventually position of optical center could be calculated using trilateration. The robot's orientation is then computed based on the camera model and landmark positions.

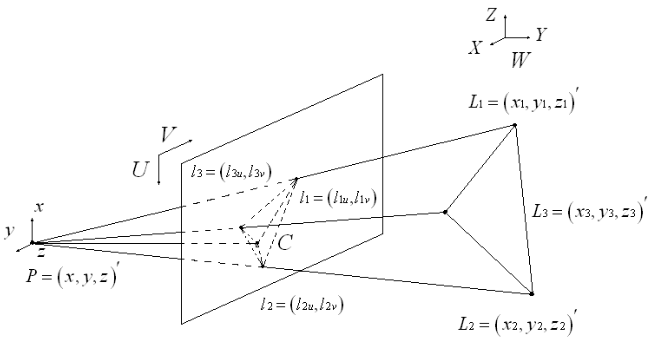


Figure 3.1: Pinhole camera model

3.1.1 Visual Angle Estimation

Based on the pinhole camera model (Figure 3.1), P represents the optical center, xyz denotes the camera frame, UV denotes the image plane, and C denotes the image center of UV. In principle, the optical axis PC is perpendicular to the plane UV, and the length of PC is the focal length f . In addition, L_i, where $i = (1,2,3)$, denotes the ith landmark with a known position defined in the world frame XYZ, and l_i denotes the projection of L_i in UV.

The camera model follows the principle of perspective projection. A detailed description of camera parameters and estimation algorithms can be found in a computer vision book such as [44]. A standard calibration toolbox can also be found at "http://www.vision.caltech.edu/bouguetj/calib_doc/".

Assuming that the camera is fixed on the mobile robot, robot positioning is equivalent to finding the position of the optical center $P = (x, y, z)'$. Besides, the optical axis PC (Figure 3.1) is chosen to represent the robot orientation in our method. We assume that the image center **C** and focal length f are known (in practice they can be obtained from camera calibration), and:

$\| PC \| = f$, $C = (c_u - c_v)'$, $l_1 = (l_{1u}, l_{1v})'$, $l_2 = (l_{2u}, l_{2v})'$, $l_3 = (l_{3u}, l_{3v})'$, $L_1 = (x_1, y_1, z_1)'$, $L_2 = (x_2, y_2, z_2)', L_3 = (x_3, y_3, z_3)'$

In the image frame, all the lengths are measured in the unit of pixel. Since **PC** is perpendicular to the image plane **UV**,

$$\begin{aligned} \| Pl_i \| &= \sqrt{f^2 + (l_{iu} - c_u)^2 + (l_{iv} - c_v)^2} \\ \| l_i l_j \| &= \sqrt{(l_{iu} - l_{ju})^2 + (l_{iv} - l_{jv})^2} \end{aligned} \tag{3.1}$$

where $i = (1,2,3)$, $j = (1,2,3)$ and $i \neq j$.

Applying the law of cosine to the triangle $\mathbf{l_i P l_j}$, we obtain

$$\angle l_i P l_j = \cos^{-1}\left(\frac{\| Pl_i \|^2 + \| Pl_j \|^2 - \| l_i l_j \|^2}{2 \| Pl_i \|| \cdot \| Pl_j |} \right) \tag{3.2}$$

Moreover, in the pinhole camera model, the visual angle between landmarks i and j, $\angle L_i P L_j$, is the same angle between their projections, $\angle l_i P l_j$. For the convenience of expression, we define $\phi_{12} = \angle L_1 P L_2 = \angle l_1 P l_2$ as the visual angle between landmarks 1 and 2. Visual angles ϕ_{13} and ϕ_{23} are defined similarly.

3.1.2 Position Estimation

Applying the law of cosine to the triangles L_1PL_2 , L_1PL_3 ,and L_2PL_3 respectively, we have

$$\begin{aligned} &\| PL_1 \|^2 + \| PL_2 \|^2 - 2\cos\phi_{12} \| PL_1 \| \cdot \| PL_2 \| - \| L_1 L_2 \|^2 = 0 \\ &\| PL_1 \|^2 + \| PL_3 \|^2 - 2\cos\phi_{13} \| PL_1 \| \cdot \| PL_3 \| - \| L_1 L_3 \|^2 = 0 \\ &\| PL_2 \|^2 + \| PL_3 \|^2 - 2\cos\phi_{23} \| PL_2 \| \cdot \| PL_3 \| - \| L_2 L_3 \|^2 = 0 \end{aligned} \tag{3.3}$$

where $\| L_i L_j \|$ denotes the known distance between landmarks i and j, $\| L_i L_j \| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$, and $\| PL_i \|$ denotes the unknown distance between P and Li.

Newton's method [46] is used to compute $\| PL_i \|$ in Equ.3. Once the distances $\| PL_i \|$ are obtained, the optical center can be located by solving a trilateration problem,

$$\begin{aligned} &(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = \| PL_1 \|^2 \\ &(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = \| PL_2 \|^2 \\ &(x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = \| PL_3 \|^2 \end{aligned} \tag{3.4}$$

Both closed and iterative form solutions are available [1-4].

To sum up, the distances between the optical center and the landmarks are calculated from the visual angles and landmark positions using the law of cosine, and the robot position is estimated by trilateration. During this procedure, only the landmark positions and projections are used as input. No direct distance measurement is required.

A further inspection on (3-3) and (3-4) reveals the geometric meaning of this problem. Substituting (3-4) into (3-3), we can formulate a system of equations.

$$\begin{aligned}&\cos^2\phi_{ij}[(x-x_i)^2+(y-y_i)^2+(z-z_i)^2][(x-x_j)^2+(y-y_j)^2+(z-z_j)^2]\\&=[x^2+y^2+z^2-x(x_i+x_j)-y(y_i+y_j)-z(z_i+z_j)+x_ix_j+y_iy_j+z_iz_j]\end{aligned} \tag{3.5}$$

3.1.3 Orientation Estimation

In a 3D environment, an orientation can be represented by a directional vector. Since the camera is fixed on the mobile robot, finding the robot orientation is equivalent to finding the camera orientation. Therefore, the optical axis is chosen to represent the robot orientation.

In the world frame XYZ, The lines **PC** and **PL$_i$** can be expressed as following

$$\text{Line PC: } \frac{x-x_p}{p_x}=\frac{y-y_p}{p_y}=\frac{z-z_p}{p_z} \tag{3.6}$$

$$\text{Line PL}_i\text{: } \frac{x-x_p}{x_i-x_p}=\frac{y-y_p}{y_i-y_p}=\frac{z-z_p}{z_i-z_p} \tag{3.7}$$

where $(x_p, y_p, z_p)'$ is the estimated robot position **P** (the position of the camera optical center), $(x_i, y_i, z_i)'$ is the global position of landmark **L$_i$** , and $(p_x, p_y, p_z)'$ is the directional vector of **PC** which needs to be determined.

Corresponding to (3-6) and (3-7), the angle between PC and PL$_i$ satisfies the following relationship:

$$\phi_{ic}=\cos^{-1}\frac{p_x(x_i-x_p)+p_y(y_i-y_p)+p_z(z_i-z_p)}{\sqrt{p_x^2+p_y^2+p_z^2}\sqrt{(x_i-x_p)^2+(y_i-y_p)^2+(z_i-z_p)^2}} \tag{3.8}$$

By normalizing the directional vector **PC**, we have $p_x{}^2 + p_y{}^2 + p_z{}^2 = 1$, and (3-8) can be rewritten as:

$$\phi_{ic} = \cos^{-1}\frac{p_x(x_i - x_p) + p_y(y_i - y_p) + p_z(z_i - z_p)}{\sqrt{(x_i - x_p)^2 + (y_i - y_p)^2 + (z_i - z_p)^2}} \tag{3.9}$$

On the other hand, we can calculate ϕ_{ic} in camera's frame:

$$\phi_{ic} = \cos^{-1}\left(\frac{\| PL_i \|^2 + \| PC \|^2 - \| L_i C \|^2}{2\| PL_i \| \cdot \| PC \|}\right) = \cos^{-1}\frac{f}{\| PL_i \|} \tag{3.10}$$

where we have used the facts that $\| PL_i \| = \sqrt{f^2 + (l_{iu} - c_u)^2 + (l_{iv} - c_v)^2}$, $\| PC \| = f$, and $\| L_i c \| = \sqrt{(l_{iu} - c_u)^2 + (l_{iv} - c_v)^2}$. The quantities l_{iu}, l_{iv}, c_u and c_v can be obtained directly from the image while f can be obtained from camera calibration.

According to (3-9) and (3-10), we have

$$p_x(x_i - x_p) + p_y(y_i - y_p) + p_z(z_i - z_p) = \frac{f\sqrt{(x_i - x_p)^2 + (y_i - y_p)^2 + (z_i - z_p)^2}}{\sqrt{f^2 + (l_{iu} - c_u)^2 + (l_{iv} - c_v)^2}} \tag{3.11}$$

The solution of (3-11), with $i = (1,2,3)$, can be written in the following matrix form:

$$PC = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} x_1 - x_p & y_1 - y_p & z_1 - z_p \\ x_2 - x_p & y_2 - y_p & z_2 - z_p \\ x_3 - x_p & y_3 - y_p & z_3 - z_p \end{pmatrix}^{-1} \cdot \begin{pmatrix} \dfrac{f\sqrt{(x_1 - x_p)^2 + (y_1 - y_p)^2 + (z_1 - z_p)^2}}{\sqrt{f^2 + (l_{1u} - c_u)^2 + (l_{1v} - c_v)^2}} \\ \dfrac{f\sqrt{(x_2 - x_p)^2 + (y_2 - y_p)^2 + (z_2 - z_p)^2}}{\sqrt{f^2 + (l_{2u} - c_u)^2 + (l_{2v} - c_v)^2}} \\ \dfrac{f\sqrt{(x_3 - x_p)^2 + (y_3 - y_p)^2 + (z_3 - z_p)^2}}{\sqrt{f^2 + (l_{3u} - c_u)^2 + (l_{3v} - c_v)^2}} \end{pmatrix} \tag{3.12}$$

which gives the direction of the optical axis PC (equivalent to the robot orientation).

3.2 Error Analysis

In order to verify real experimental error of the algorithm, an experiment is carried out. The general objective of this experiment is to verify algorithm accuracy under different circumstances, for example, how accuracy changes with different distances from robot to landmarks, how distribution of landmarks affects overall accuracy is also studied.

The most difficult part in most experiments is that it is impossible to get true values that the data generated from algorithm can be compared with. In most cases, instead, an experiment is designed to obtain the data that is mostly equal to the true values.

In our experiment, a check board pattern, which is commonly used in camera calibration, is used to obtain robot’s ideal position and orientation with respect to the pattern coordinate system (which the world frame is attached with).

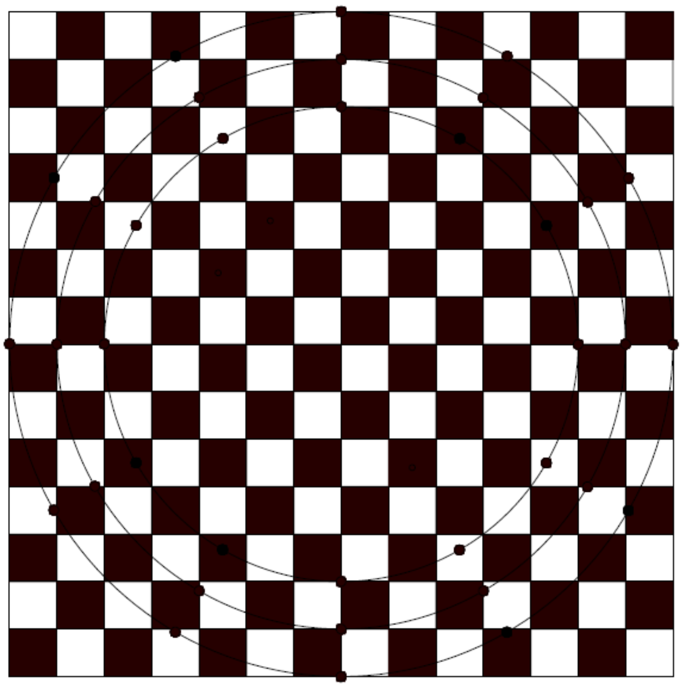


Figure 3.2: Check board pattern printed together with three circles

Another objective of the experiment is to test accuracy with robot placed at different directions of those landmarks but with absolute the same distance from the pattern, only in this way do the results really reflect accuracy changes with different directions. The intuitive way of doing this is to manually place robot to different directions of the pattern. However, once robot is moved, it is impossible to keep the robot the same distance from the pattern. Therefore, another experiment design came into our mind, that is, is it possible that we rotate the pattern instead of moving robot? A second thought reminds us that this scheme also has a problem, that is, it is very hard to make pattern rotate absolutely with respect to its center. Eventually, we designed our current experiment. Three concentric circles are drawn in the same board with check board pattern, and each circle has 12 landmarks on it (Figure 3.2). Instead of moving robot after every test, different groups of landmarks are used every time. Moreover, by using different groups of landmarks, the board doesn't need to rotate itself. For example, in the first test, landmark 1, 5, and 9 are used, and the first group of results are obtained. Then the second group of landmarks with 2, 6 and 10 is taken into calculation to mimic the situation that the board has just rotate 30 degrees. Therefore, without moving robot or board, the whole tests could be done smoothly and with high reliability.

To sum up, the experiment is conducted in this order. Firstly, 40 pictures from different angles are taken for camera calibration using Matlab camera calibration toolbox. Camera's intrinsic parameters, which include camera's focal length, principal points, are obtained. Secondly, pictures of the check board pattern are taken when robot is placed at different distances. Thirdly, based on previously obtained intrinsic parameters and with help of Matlab's calibration toolbox, robot's position and orientation can be calculated and they are assumed to be true. Finally, robot (optical center of the camera)'s position and orientation is calculated from our localization algorithm based on landmarks. The two groups of data are compared.

3.2.1 Camera Calibration Using Matlab Toolbox

From Figure 3.3, 40 pictures are taken from different angles. By manually marking four corners of the pattern, the Matlab calibration toolbox is able to calculate camera's intrinsic parameters. The more pictures are used, the more accurate the results are. In our case, 40 pictures are taken and used. The calibration parameters are listed as following:

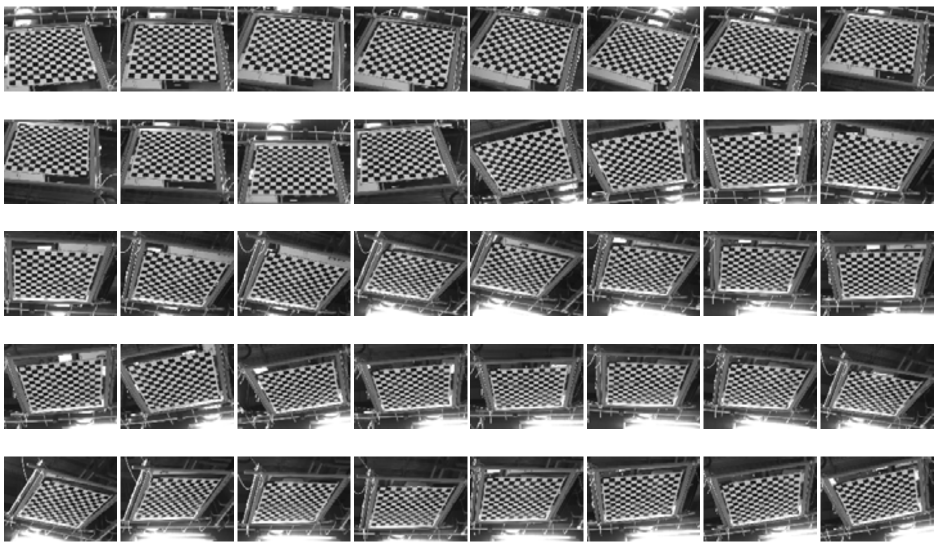


Figure 3.3: Pictures used in camera calibration

Focal length: fc=[1627.6, 1629.9],

Principal point: cc=[333.9088, 246.3799],

Skew coefficient, $\alpha_c = 0$

Distortion coefficients, kc= [-0.3350, -0.2882, 0.0007, -0.0012, 0],

Focal length uncertainty, $f_{c,error}$ = [1.1486, 1.1069],

Principal point uncertainty, $c_{c,error}$ =[1.9374, 1.5782],

Skew coefficient uncertainty, $\alpha_{c,error=0}$,

Distortion coefficients uncertainty, =[0.0084,0.1936,0.0002,0.0002,0],

Image size: 640×480.

3.2.2 Pictures Taken From Different Distances

As we know that on each circle there are 12 evenly distributed points (Figure 3.2). The basic idea behind this experiment is to use check board pattern for calculation of transform matrix between pattern frame and camera frame, therefore we can obtain robot's ideal position and orientation. The results are then compared with what is obtained using our localization algorithm based on landmarks. By using different groups of landmarks (each group has 3 landmarks from the same circle), images of the pattern are taken with robot placed at different distances from the pattern, algorithm errors from different distances with different groups of landmarks can be obtained and compared.

Design radiuses of circles are 250mm, 300mm and 350mm (Figure 3.4). However, due to print error, the real radiuses are 244mm, 292.8mm and 341.6mm. Since length error is evenly distributed and each check board grid is still exactly the same size (48.8mm×48.8mm), it doesn't affect our algorithm if we take the real values in our calculation.

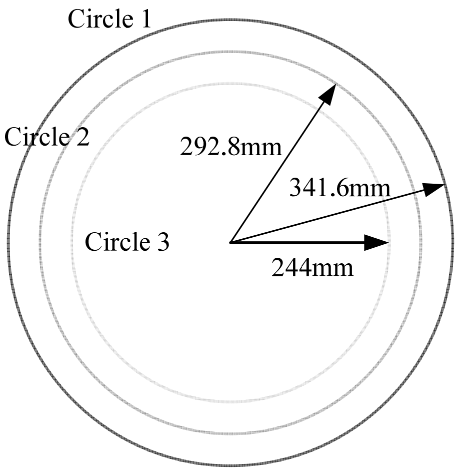


Figure 3.4: Radiuses of three circles

Initial vertical distance from the board to floor is approximately 2200mm, the camera is around 700mm above the ground.

It is noted that when pictures are being taken, pixel of the center of circles should be located in the principal point, only in this way can we assume that the pattern's center remain the same distance to camera's optical center, no matter which group of landmarks is used. This requirement is satisfied by marking principle points in images and make sure center of the pattern match that point.

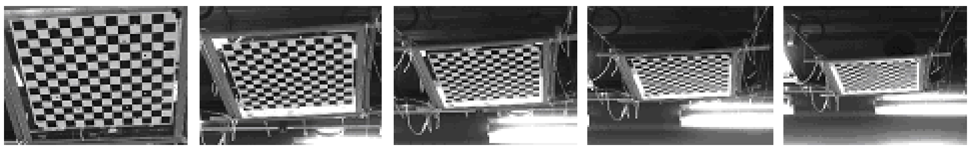


Figure 3.5: Four photos of the pattern are taken from five different distances

3.2.3 True Positions and Orientations of Robot

As we mentioned earlier that true positions and orientations of the robot at different distances are calculated by using Matlab camera calibration toolbox.

Table 3.1: Real relative position and orientation of robot when 2 meters away

Landmarks used	x (mm)	y (mm)	z (mm)	θ (Direction Vector)
1,5,9	-132.8737	-1493.7066	1549.2967	[0.0604,0.6932,-0.7182]
2,6,10	-861.9253	-1227.151	1549.2967	[0.3989,0.5701,-0.7182]
3,7,11	-1360.0247	-631.7813	1549.2967	[0.6305,0.2943,-0.7182]
4,8,12	-1493.7066	132.8737	1549.2967	[0.6932,-0.0604,-0.7182]
5,9,1	-1227.151	861.9253	1549.2967	[0.5701,-0.3989,-0.7182]
6,10,2	-631.7813	1360.0247	1549.2967	[0.2943,-0.6305,-0.7182]
7,11,3	132.8737	1493.7066	1549.2967	[-0.0604,-0.6932,-0.7182]
8,12,4	861.9253	1227.151	1549.2967	[-0.3989,-0.5701,-0.7182]
9,1,5	1360.0247	631.7813	1549.2967	[-0.6305,-0.2943,-0.7182]
10,2,6	1493.7066	-132.8737	1549.2967	[-0.6932,0.0604,-0.7182]
11,3,7	1227.151	-861.9253	1549.2967	[-0.5701,0.3989,-0.7182]
12,4,8	631.7813	-1360.0247	1549.2967	[-0.2943,0.6305,-0.7182]

Table 3.2: Real relative position and orientation of robot when 2.5 meters away

Landmarks used	x (mm)	y (mm)	z (mm)	θ (Direction Vector)
1,5,9	-236.0603	-2240.5663	1519.9805	[0.0859,0.825,-0.5586]
2,6,10	-1324.7174	-1822.3572	1519.9805	[0.4869,0.6716,-0.5586]
3,7,11	-2058.4175	-915.8489	1519.9805	[0.7574,0.3382,-0.5586]
4,8,12	-2240.5663	236.0603	1519.9805	[0.825,-0.0859,-0.5586]
5,9,1	-1822.3572	1324.7174	1519.9805	[0.6716,-0.4869,-0.5586]
6,10,2	-915.8489	2058.4175	1519.9805	[0.3382,-0.7574,-0.5586]
7,11,3	236.0603	2240.5663	1519.9805	[-0.0859,-0.825,-0.5586]
8,12,4	1324.7174	1822.3572	1519.9805	[-0.4869,-0.6716,-0.5586]
9,1,5	2058.4175	915.8489	1519.9805	[-0.7574,-0.3382,-0.5586]
10,2,6	2240.5663	-236.0603	1519.9805	[-0.825,0.0859,-0.5586]
11,3,7	1822.3572	-1324.7174	1519.9805	[-0.6716,0.4869,-0.5586]
12,4,8	915.8489	-2058.4175	1519.9805	[-0.3382,0.7574,-0.5586]

Table 3.3: Real relative position and orientation of robot when 3 meters away

Landmarks used	x (mm)	y (mm)	z (mm)	θ (Direction Vector)
1,5,9	-255.7185	-2771.846	1506.1401	[0.0794,0.876,-0.4757]
2,6,10	-1607.3817	-2272.6298	1506.1401	[0.5068,0.7189,-0.4757]
3,7,11	-2528.3483	-1164.4643	1506.1401	[0.7984,0.3692,-0.4757]
4,8,12	-2771.846	255.7185	1506.1401	[0.876,-0.0794,-0.4757]
5,9,1	-2272.6298	1607.3817	1506.1401	[0.7189,-0.5068,-0.4757]
6,10,2	-1164.4643	2528.3483	1506.1401	[0.3692,-0.7984,-0.4757]
7,11,3	255.7185	2771.846	1506.1401	[-0.0794,-0.876,-0.4757]
8,12,4	1607.3817	2272.6298	1506.1401	[-0.5068,-0.7189,-0.4757]
9,1,5	2528.3483	1164.4643	1506.1401	[-0.7984,-0.3692,-0.4757]
10,2,6	2771.846	-255.7185	1506.1401	[-0.876,0.0794,-0.4757]
11,3,7	2272.6298	-1607.3817	1506.1401	[-0.7189,0.5068,-0.4757]
12,4,8	1164.4643	-2528.3483	1506.1401	[-0.3692,0.7984,-0.4757]

Table 3.4: Real relative position and orientation of robot when 3.5 meters away

Landmarks used	x (mm)	y (mm)	z (mm)	θ (Direction Vector)
1,5,9	-222.962	-3319.2174	1502.1583	[0.0601,0.9099,-0.4105]
2,6,10	-1852.6995	-2763.0456	1502.1583	[0.507,0.7579,-0.4105]
3,7,11	-2986.0076	-1466.518	1502.1583	[0.818,0.4029,-0.4105]
4,8,12	-3319.2174	222.962	1502.1583	[0.9099,-0.0601,-0.4105]
5,9,1	-2763.0456	1852.6995	1502.1583	[0.7579,-0.507,-0.4105]
6,10,2	-1466.518	2986.0076	1502.1583	[0.4029,-0.818,-0.4105]
7,11,3	222.962	3319.2174	1502.1583	[-0.0601,-0.9099,-0.4105]
8,12,4	1852.6995	2763.0456	1502.1583	[-0.507,-0.7579,-0.4105]
9,1,5	2986.0076	1466.518	1502.1583	[-0.818,-0.4029,-0.4105]
10,2,6	3319.2174	-222.962	1502.1583	[-0.9099,0.0601,-0.4105]
11,3,7	2763.0456	-1852.6995	1502.1583	[-0.7579,0.507,-0.4105]
12,4,8	1466.518	-2986.0076	1502.1583	[-0.4029,0.818,-0.4105]

Table 3.5: Real relative position and orientation of robot when 4 meters away

Landmarks used	x (mm)	y (mm)	z (mm)	θ (Direction Vector)
1,5,9	-223.0791	-3899.0654	1472.9688	[0.0519,0.9351,-0.3505]
2,6,10	-2142.7249	-3265.1501	1472.9688	[0.5125,0.7839,-0.3505]
3,7,11	-3488.2292	-1756.3405	1472.9688	[0.8358,0.4227,-0.3505]
4,8,12	-3899.0654	223.0791	1472.9688	[0.9351,-0.0519,-0.3505]
5,9,1	-3265.1501	2142.7249	1472.9688	[0.7839,-0.5125,-0.3505]
6,10,2	-1756.3405	3488.2292	1472.9688	[0.4227,-0.8358,-0.3505]
7,11,3	223.0791	3899.0654	1472.9688	[-0.0519,-0.9351,-0.3505]
8,12,4	2142.7249	3265.1501	1472.9688	[-0.5125,-0.7839,-0.3505]
9,1,5	3488.2292	1756.3405	1472.9688	[-0.8358,-0.4227,-0.3505]
10,2,6	3899.0654	-223.0791	1472.9688	[-0.9351,0.0519,-0.3505]
11,3,7	3265.1501	-2142.7249	1472.9688	[-0.7839,0.5125,-0.3505]
12,4,8	1756.3405	-3488.2292	1472.9688	[-0.4227,0.8358,-0.3505]

3.2.4 Robot's Estimated Position and Orientation Using Our Localization Algorithm

Camera calibration parameters are imported to our program and rectified images are input. The positions and orientations of the robot are calculated with robot placed at different distances and different groups of landmarks are used. It doesn't have enough space to list all data here, the comparison results, which refers to inconsistency at x, y z axis and angle, are listed from Table 3.6 to 3.10. Standard deviations of these errors are also listed in Table 3.11 and draw in Figure 3.6. Experimental study using 4 landmarks are also conducted and the results are shown in from Table 3.12 to Table 3.17, Figure 3.7 draws the standard deviations.

Table 3.6: 2 meters way

Circle	1				2				3			
Error	E_x (mm)	E_y (mm)	E_z (mm)	E_θ (degree)	E_x (mm)	E_y (mm)	E_z (mm)	E_θ (degree)	E_x (mm)	E_y (mm)	E_z (mm)	E_θ (degree)
1,5,9	2.3589	0.5639	2.3523	0.12095	16.123	1.8182	-1.072	0.50949	10.916	-6.5477	-11.831	0.48561
2,6,10	5.4365	1.6032	1.3473	0.20598	2.8041	-5.8212	-2.3867	0.21998	7.7278	-2.3149	2.1165	0.25956
3,7,11	1.6336	14.138	2.14	0.39564	-6.9758	5.1213	-8.5319	0.27524	-15.995	-6.0953	-18.167	0.69636
4,8,12	-8.2808	0.55243	-11.255	0.34783	-4.0289	1.6254	-9.0608	0.20948	-7.8789	-1.6559	-15.071	0.44005
5,9,1	-0.6911	-2.3248	2.3523	0.12095	-6.4868	-14.872	-1.072	0.50949	-11.129	-6.18	-11.831	0.48561
6,10,2	-1.3298	-5.5097	1.3473	0.20598	-6.4433	0.48212	-2.3867	0.21998	-5.8687	-5.5351	2.1165	0.25956
7,11,3	11.427	-8.4838	2.14	0.39564	7.9231	3.4806	-8.5319	0.27524	2.7186	16.899	-18.167	0.69636
8,12,4	4.6188	6.8952	-11.255	0.34783	3.4221	2.6764	-9.0608	0.20948	2.5054	7.6513	-15.071	0.44005
9,1,5	-1.6678	1.7609	2.3523	0.12095	-9.6361	13.054	-1.072	0.50949	0.2122	12.728	-11.831	0.48561
10,2,6	-4.1066	3.9065	1.3473	0.20598	3.6392	5.339	-2.3867	0.21998	-1.8592	7.8499	2.1165	0.25956
11,3,7	-13.061	-5.6543	2.14	0.39564	-0.94726	-8.6019	-8.5319	0.27524	13.276	-10.804	-18.167	0.69636
12,4,8	3.662	-7.4476	-11.255	0.34783	0.60681	-4.3018	-9.0608	0.20948	5.3735	-5.9953	-15.071	0.44005

Table 3.7: 2.5 meters way

Circle	1				2				3			
Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ

1,5,9	-8.0787	5.7478	9.3783	0.30631	0.21915	2.7175	-6.8621	0.06908	9.7288	2.1814	-2.4295	0.21974
2,6,10	7.2372	2.8902	-1.0217	0.1396	3.3917	0.52986	-8.5322	0.15313	0.6158	-2.6668	0.93699	0.06668
3,7,11	12.063	-6.958	-6.5385	0.29515	-1.9442	13.985	-2.4959	0.28204	-8.2195	13.626	-11.423	0.42614
4,8,12	2.5461	8.111	3.9832	0.20101	1.3882	-2.2092	-6.6392	0.10511	-1.4439	6.6614	-11.635	0.26146
5,9,1	9.0171	4.1225	9.3783	0.30631	2.2438	-1.5485	-6.8621	0.06908	-2.9752	-9.5161	-2.4295	0.21974
6,10,2	-1.1156	-7.7128	-1.0217	0.1396	-1.237	-3.2023	-8.5322	0.15313	-2.6175	0.80003	0.93699	0.06668
7,11,3	-12.057	-6.9676	-6.5385	0.29515	13.083	-5.3087	-2.4959	0.28204	15.91	0.30537	-11.423	0.42614
8,12,4	5.7513	-6.2605	3.9832	0.20101	-2.6073	-0.0976	-6.6393	0.10511	6.4909	-2.0802	-11.635	0.26146
9,1,5	0.93837	-9.8703	9.3783	0.30631	-2.463	-1.1689	-6.8621	0.06908	-6.7536	7.3347	-2.4295	0.21974
10,2,6	-6.1216	4.8225	-1.0217	0.1396	-2.1547	2.6724	-8.5322	0.15313	2.0016	1.8668	0.93698	0.06668
11,3,7	0.00553	13.926	-6.5385	0.29515	-11.139	-8.6761	-2.4959	0.28204	-7.6906	-13.931	-11.423	0.42614
12,4,8	-8.2973	-1.8505	3.9832	0.20101	1.2191	2.3068	-6.6393	0.10511	-5.047	-4.5812	-11.635	0.26146

Table 3.8: 3 meters way

Circle	1				2				3			
Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
1,5,9	8.2371	17.013	-4.5953	0.19698	12.778	6.3152	-6.9821	0.28376	29.126	9.0809	-20.716	0.6598
2,6,10	-6.1736	10.969	0.11138	0.23224	13.422	9.9015	-8.8295	0.15535	10.572	-3.2921	-3.4803	0.22778
3,7,11	19.159	10.081	-1.4253	0.22004	18.467	-0.3134	0.79374	0.29121	10.072	3.1388	-23.418	0.29212
4,8,12	4.2165	-0.3756	-14.214	0.18638	11.798	-9.6742	-22.481	0.30452	10.552	-13.581	-38.6	0.5902
5,9,1	10.615	-15.64	-4.5953	0.19698	-0.92003	-14.224	-6.9821	0.28376	-6.6989	-29.765	-20.716	0.6598
6,10,2	12.586	-0.13777	0.11138	0.23224	1.8642	-16.574	-8.8295	0.15535	-8.1371	-7.5098	-3.4803	0.22778
7,11,3	0.84885	-21.633	-1.4253	0.22004	-9.505	-15.836	0.79374	0.29121	-2.3179	-10.292	-23.418	0.29212
8,12,4	-2.4335	-3.4638	-14.214	0.18638	-14.277	-5.3802	-22.481	0.30452	-17.037	-2.3484	-38.6	0.5902
9,1,5	-18.852	-1.373	-4.5953	0.19698	-11.858	7.9087	-6.9821	0.28376	-22.427	20.684	-20.716	0.6598
10,2,6	-6.4122	-10.831	0.11138	0.23224	-15.286	6.6727	-8.8295	0.15535	-2.4351	10.802	-3.4803	0.22778
11,3,7	-18.31	11.552	-1.4253	0.22004	-8.9622	16.15	0.79374	0.29121	-7.7544	7.1535	-23.418	0.29212
12,4,8	-1.7829	3.8394	-14.214	0.18638	2.4792	15.054	-22.481	0.30452	6.4849	15.929	-38.6	0.5902

Table 3.9: 3.5 meters way

Circle	1	2	3

Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
1,5,9	-15.657	35.633	8.0942	0.49878	-15.125	12.533	1.522	0.30135	11.026	19.234	-12.48	0.17405
2,6,10	0.98953	7.8926	-11.697	0.13627	7.1639	23.291	0.87403	0.23166	-11.823	28.841	5.8916	0.48857
3,7,11	23.753	32.36	16.289	0.60536	12.79	37.184	-4.6346	0.4921	11.939	16.611	-29.687	0.35372
4,8,12	12.252	3.5262	-11.335	0.11326	20.426	4.9037	-15.578	0.1209	26.624	-9.8499	-7.997	0.16266
5,9,1	38.687	-4.2573	8.0942	0.49878	18.417	6.8317	1.522	0.30135	11.144	-19.166	-12.48	0.17405
6,10,2	6.3405	-4.8033	-11.697	0.13627	16.588	-17.849	0.87402	0.23165	30.889	-4.1812	5.8916	0.48857
7,11,3	16.149	-36.751	16.289	0.60536	25.807	-29.668	-4.6346	0.4921	8.4162	-18.645	-29.687	0.35372
8,12,4	-3.0724	-12.374	-11.335	0.11326	-5.9665	-20.142	-15.578	0.1209	-21.842	-18.132	-7.997	0.16266
9,1,5	-23.031	-31.375	8.0942	0.49878	-3.2919	-19.365	1.522	0.30135	-22.17	-0.0685	-12.48	0.17405
10,2,6	-7.33	-3.0894	-11.697	0.13627	-23.752	-5.4411	0.87402	0.23166	-19.066	-24.66	5.8916	0.48857
11,3,7	-39.901	4.3902	16.289	0.60536	-38.597	-7.5157	-4.6346	0.4921	-20.355	2.034	-29.687	0.35372
12,4,8	-9.1799	8.8478	-11.335	0.11326	-14.46	15.238	-15.578	0.1209	-4.7818	27.982	-7.997	0.16266

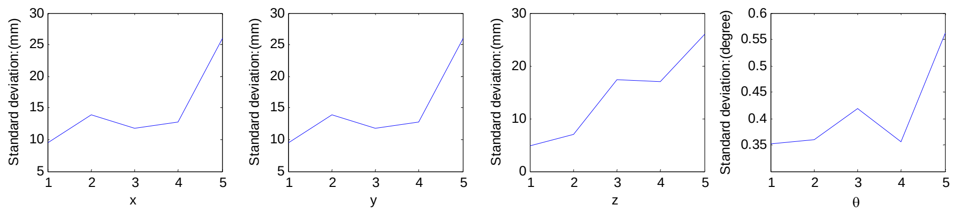
Table 3.10: 4 meters way

Circle	1				2				3			
Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
1,5,9	35.073	-17.65	7.132	0.51387	44.934	-6.2662	1.0455	0.65838	8.3944	5.0895	-1.5009	0.12583
2,6,10	20.153	-23.231	1.1207	0.4291	-34.588	-10.785	37.562	0.46747	2.7996	-28.096	10.936	0.28589
3,7,11	2.5609	-19.555	11.884	0.31389	14.556	13.667	-17.75	0.16439	-32.19	23.335	-17.126	0.59922
4,8,12	-7.7941	-27.202	4.5052	0.42811	-23.244	20.012	63.927	0.77213	-24.828	-35.116	-1.3092	0.57307
5,9,1	-32.822	-21.549	7.132	0.51387	-27.893	-35.78	1.0455	0.65838	0.2105	-9.8145	-1.5009	0.12583
6,10,2	-30.195	-5.8377	1.1207	0.4291	7.954	35.346	37.562	0.46747	-25.732	11.623	10.936	0.28589
7,11,3	-18.216	7.5599	11.884	0.31389	4.5576	-19.439	-17.75	0.16439	36.304	16.21	-17.126	0.59922
8,12,4	-19.661	20.351	4.5052	0.42811	28.953	10.124	63.927	0.77213	-17.998	39.06	-1.3092	0.57307
9,1,5	-2.2513	39.2	7.132	0.51387	-17.04	42.047	1.0455	0.65838	-8.6048	4.725	-1.5009	0.12583
10,2,6	10.042	29.068	1.1207	0.4291	26.634	-24.561	37.562	0.46747	22.932	16.472	10.936	0.28589
11,3,7	15.655	11.996	11.884	0.31389	-19.114	5.7726	-17.75	0.16439	-4.1135	-39.545	-17.126	0.59922
12,4,8	27.455	6.8512	4.5052	0.42811	-5.7084	-30.136	63.927	0.77213	42.825	-3.9433	-1.3092	0.57307

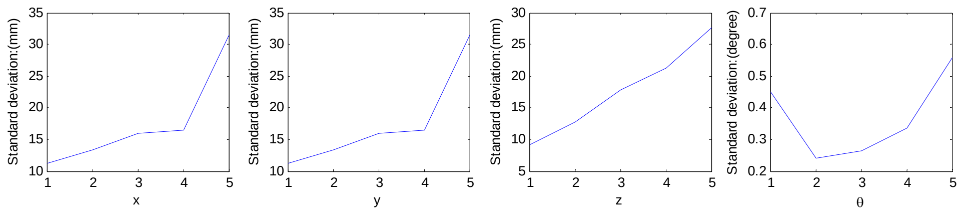
Table 3.11: Summary of standard deviations

Circle	1	2	3

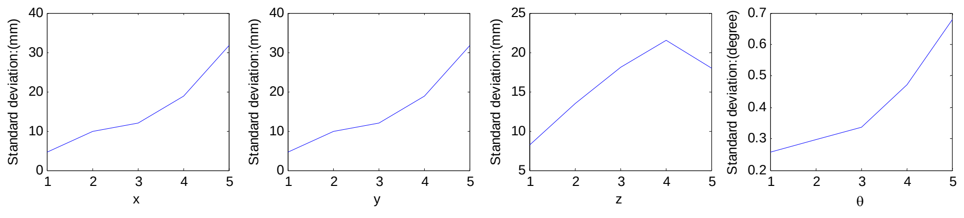
Error	E_x (mm)	E_y (mm)	E_z (mm)	E_θ (degree)	E_x (mm)	E_y (mm)	E_z (mm)	E_θ (degree)	E_x (mm)	E_y (mm)	E_z (mm)	E_θ (degree)
2	9.5183	9.5183	4.8343	0.35139	11.302	11.302	9.2476	0.44897	4.5412	4.5412	8.2613	0.25596
2.5	13.937	13.937	6.8697	0.35906	13.378	13.378	12.712	0.24136	9.8258	9.8258	13.506	0.29804
3	11.825	11.825	17.319	0.41932	15.921	15.921	17.774	0.26521	11.991	11.991	18.029	0.33762
3.5	12.681	12.681	16.877	0.35495	16.419	16.419	21.316	0.33417	18.942	18.942	21.47	0.46989
4	25.944	25.944	25.902	0.5624	31.528	31.528	27.579	0.55922	31.721	31.721	17.936	0.67912



(a) With landmarks on circle 1



(b) With landmarks on circle 2



(c) With landmarks on circle 3

Figure 3.6: Standard deviations

Besides 3 landmarks, error analysis with 4 landmarks is conducted and the results are listed in the following tables.

Table 3.12: 2 meters away

Circle	1				2				3			
Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
1,5,9	-1.758	10.3451	-7.8924	0.1053	-5.1218	5.7323	0.502	0.1962	-2.7477	1.1181	-2.613	0.0819
2,6,10	-9.4911	13.5189	0.3281	0.4289	0.2328	14.244	3.6386	0.3753	-8.0938	0.1095	-3.0529	0.3532
3,7,11	-2.4012	16.2551	4.0071	0.472	0.2533	12.0821	-9.0388	0.2681	2.0772	2.4702	10.4478	0.2675
4,8,12	10.3451	1.7581	-7.8926	0.1053	5.732	5.1222	0.5006	0.1962	1.1181	2.7479	-2.6131	0.0819
5,9,1	13.5189	9.4918	0.3272	0.4289	14.2384	-0.244	3.6427	0.3753	0.1072	8.1008	13.0681	0.3536
6,10,2	16.2551	2.401	4.0074	0.472	12.0798	-0.252	-9.046	0.2681	2.4712	-2.0762	10.4475	0.2675
7,11,3	1.7581	-10.3451	-7.8926	0.1053	5.1219	-5.7324	0.502	0.1962	2.7481	-1.1178	-2.6144	0.0819
8,12,4	9.4904	-13.519	0.3295	0.4289	-0.2494	-14.2533	3.681	0.3762	8.0849	-0.1129	-3.0299	0.3526
9,1,5	2.4009	-16.2551	4.0074	0.472	-0.2529	-12.0825	-9.0388	0.2681	-2.0767	-2.4709	10.4497	0.2675
10,2,6	-0.3451	-1.7582	-7.8927	0.1053	-5.7322	-5.1218	0.5018	0.1962	-1.118	-2.7477	-2.6132	0.0819
11,3,7	-3.5185	-9.4914	0.3268	0.4289	-14.2432	0.2396	3.6443	0.3754	-0.1077	-8.1016	-3.0645	0.3535
12,4,8	-6.2549	-2.4013	4.0063	0.472	-12.08	0.2499	-9.0489	0.2681	-2.4697	2.077	10.4487	0.2675

Table 3.13: 2.5 meters away

Circle	1	2	3

Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
1,5,9	4.7172	16.6581	2.3244	0.2588	0.7631	12.1048	-8.5656	0.0146	-5.6797	10.8691	-0.5018	0.1695
2,6,10	-1.301	25.1231	4.9753	0.4822	10.0576	18.8214	-3.0027	0.0846	-3.1164	8.1503	-8.3763	0.2903
3,7,11	9.6413	11.98	1.6002	0.2117	18.2163	-0.7453	-7.6984	0.1924	9.9699	-6.061	-1.4574	0.2248
4,8,12	16.6582	-4.7172	2.3237	0.2588	12.1049	-0.7625	-8.5656	0.0146	10.867	5.6754	-0.4945	0.1695
5,9,1	25.1228	1.3013	4.9743	0.4822	18.8236	-10.0465	-13.045	0.0843	8.1539	3.1143	-8.3216	0.2895
6,10,2	11.9605	-9.6463	1.5584	0.2109	-0.7491	-18.217	-7.7151	0.1924	-6.082	-9.9782	-1.5015	0.2256
7,11,3	-4.7172	-16.6581	2.3243	0.2588	-0.7622	-12.1039	-8.5686	0.0145	5.6757	-0.8673	-0.4963	0.1695
8,12,4	1.301	-25.1231	4.9753	0.4822	-10.0576	-18.8214	-3.0027	0.0846	3.1164	-8.1503	-8.3763	0.2903
9,1,5	-9.6468	-11.9596	1.5551	0.2108	-18.2167	0.747	-7.7051	0.1924	-9.9695	6.077	-1.5085	0.2255
10,2,6	-6.6582	4.717	2.3245	0.2588	-12.105	0.7628	-8.568	0.0145	-10.867	-5.6754	-0.4944	0.1695
11,3,7	-25.122	-1.3011	4.9741	0.4822	-18.8222	10.0505	-3.0264	0.0844	-8.1514	-3.1167	-8.3723	0.2903
12,4,8	-1.9448	9.6506	1.5305	0.2103	0.7496	18.2166	-7.7218	0.1924	6.0728	9.9725	-1.4829	0.2252

Table 3.14: 3 meters away

Circle	1				2				3			
Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
1,5,9	9.2554	16.6739	21.5476	0.5057	6.4364	8.0622	-0.4032	0.1263	-6.536	12.3671	20.8448	0.4657
2,6,10	7.632	15.7263	14.4388	0.3729	21.8249	23.846	-5.6884	0.0856	-0.7341	24.901	-9.0946	0.2923
3,7,11	2.8173	11.4115	-1.6223	0.1747	12.4404	12.5993	-21.953	0.2278	7.0791	-2.9651	-1.5461	0.155
4,8,12	16.6736	-9.2552	21.5474	0.5057	8.0622	-6.4361	-0.4038	0.1263	12.369	6.5313	20.8488	0.4657
5,9,1	15.7184	-7.6352	14.4698	0.3733	23.8453	-21.8246	-15.681	0.0856	24.905	0.7406	-9.1654	0.2929
6,10,2	11.4128	-2.8143	-1.604	0.1748	12.5917	-12.4507	-1.9659	0.2278	-2.9675	-7.0874	-1.5112	0.1545
7,11,3	-9.2555	-16.674	21.5478	0.5057	-6.4363	-8.0623	-0.4039	0.1263	6.5365	-2.3686	20.8391	0.4656
8,12,4	-7.633	-15.7195	14.4235	0.3726	-21.827	-23.8448	-5.6891	0.0856	0.7441	-4.9073	-9.1508	0.2929
9,1,5	-2.816	-11.4117	-1.6135	0.1748	-12.4423	-12.5982	-1.9542	0.2278	-7.0791	2.9651	-1.5463	0.155
10,2,6	-16.674	9.2554	21.5477	0.5057	-8.0622	6.4363	-0.4038	0.1263	-2.3691	-6.5372	20.8398	0.4656
11,3,7	-15.718	7.6353	14.469	0.3733	-23.842	21.8305	-15.661	0.0859	-4.8995	-0.7322	-9.1062	0.2923
12,4,8	-1.4135	2.8142	-1.5773	0.1749	-12.6058	12.4371	-1.9099	0.2272	2.9527	7.0716	-1.4761	0.1539

Table 3.15: 3.5 meters away

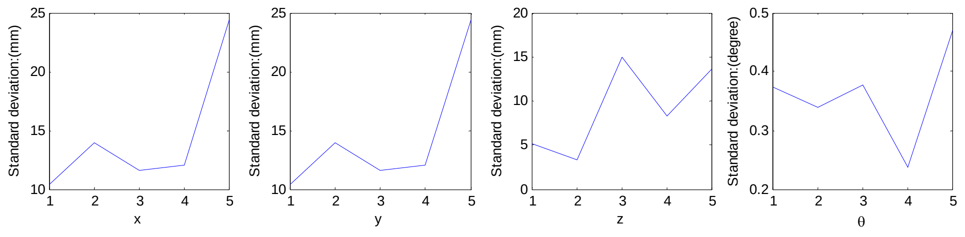
Circle	1				2				3			
Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
1,5,9	-3.7506	16.6903	5.9739	0.2069	-4.5698	6.2434	-6.864	0.088	25.4035	9.1058	14.0537	0.4613
2,6,10	13.8345	14.2228	5.8792	0.2411	15.4063	15.6661	2.3059	0.2129	26.8478	-2.032	-0.8553	0.3476
3,7,11	13.0142	-4.4684	11.6744	0.2616	2.6168	24.3667	-9.7729	0.3711	14.3337	-4.7069	-3.9566	0.3264
4,8,12	16.6911	3.7505	5.9715	0.2068	6.2442	4.5702	-6.8643	0.088	9.1058	-5.4036	14.0535	0.4613
5,9,1	14.2228	-13.8346	5.879	0.2411	15.7137	-15.39	2.1483	0.2108	-2.003	-6.8273	-1.0162	0.3474
6,10,2	-4.4711	-13.0193	11.6623	0.2615	24.3646	-2.6191	-9.7839	0.3711	-4.7128	-4.3408	-3.9938	0.3269
7,11,3	3.7506	-16.6912	5.9717	0.2068	4.57	-6.245	-6.8632	0.088	-5.4035	-9.1059	14.0534	0.4613
8,12,4	-3.8347	-14.2229	5.878	0.2411	-15.3965	-15.7028	2.1455	0.2107	-6.8342	2.0043	-1.0045	0.3475
9,1,5	-3.0205	4.4721	11.668	0.2616	-2.6189	-24.3644	-19.783	0.3711	-4.3411	4.7134	-3.9867	0.3268
10,2,6	-6.6915	-3.7508	5.9708	0.2068	-6.2447	-4.5697	-6.8627	0.088	-9.1058	25.4035	14.0538	0.4614
11,3,7	-4.2225	13.8349	5.8801	0.2411	-15.6779	15.4029	2.3289	0.2132	2.0178	26.8374	-0.9337	0.3475
12,4,8	4.4671	13.0161	11.6803	0.2617	-24.3667	2.6168	-9.7728	0.3711	4.7086	14.335	-23.957	0.3264

Table 3.16: 4 meters away

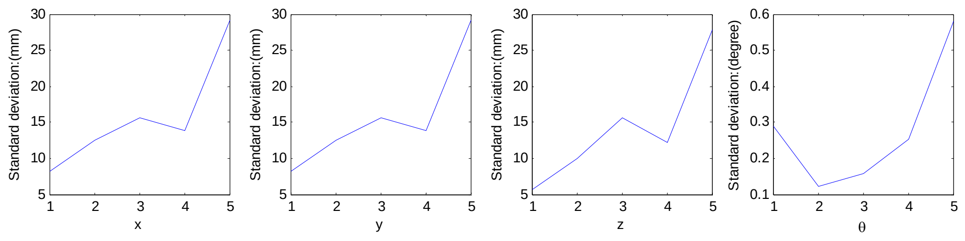
Circle	1				2				3			
Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
1,5,9	35.4853	6.7436	-15.8298	0.5417	20.8644	13.4147	-12.8503	0.3377	45.81	-3.2013	-0.1092	0.66
2,6,10	31.2703	1.1294	13.3179	0.3773	63.9709	-0.6741	-28.9919	0.7973	37.9122	-1.9537	-9.2286	0.5867
3,7,11	25.9205	-24.8835	11.298	0.475	13.0497	16.0023	-36.3094	0.5125	17.9025	-5.3291	-4.6888	0.6319
4,8,12	6.7436	-35.4853	-15.83	0.5417	13.4163	-20.8687	-12.8435	0.3377	-3.2006	-5.8142	-0.0985	0.66
5,9,1	1.1346	-31.2703	13.2897	0.3771	-0.6658	-63.9698	-29.0438	0.7974	-1.8755	-7.9068	-9.6885	0.5885
6,10,2	-4.9026	-25.9247	11.2934	0.4752	16.0017	-13.0464	-36.3011	0.5124	-35.348	-7.9169	-4.7327	0.6325
7,11,3	-5.4886	-6.7456	-15.8258	0.5418	-20.8649	-13.4111	-12.8425	0.3377	-5.8104	3.2037	-0.1037	0.66
8,12,4	-1.2706	-1.1293	13.3213	0.3773	-63.9719	0.6772	-29.0008	0.7973	-7.9193	11.9712	-19.282	0.5872
9,1,5	-5.9261	24.8928	11.3006	0.4751	-13.0485	-15.9937	-36.3153	0.5126	-7.9106	35.3423	-14.676	0.632
10,2,6	-6.7425	35.4855	-15.8273	0.5417	-13.417	20.8644	-12.8414	0.3377	3.2006	45.8138	-0.0996	0.66
11,3,7	-1.1317	31.271	13.3082	0.3772	0.676	63.975	-28.9974	0.7973	11.9662	37.9277	-9.2338	0.5869
12,4,8	24.8838	25.9207	11.2979	0.475	-16.0022	13.0493	-36.3083	0.5125	35.363	17.9456	-4.8442	0.6334

Table 3.17: Summary of standard deviations

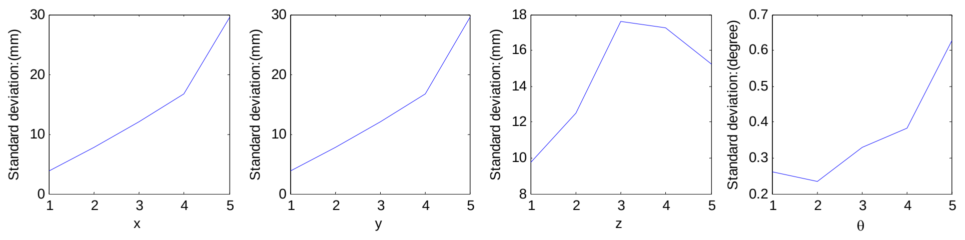
Circle	1				2				3			
Error	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ	E_x	E_y	E_z	E_θ
2	10.4318	10.432	5.1139	0.3732	8.2456	8.2485	5.6381	0.2895	3.7565	3.7607	9.7708	0.2601
2.5	13.955	13.9581	3.2958	0.3386	12.4831	12.4815	10.0389	0.1216	7.7764	7.776	12.5081	0.2336
3	11.5988	11.5998	15.0076	0.3766	15.6249	15.6261	15.5739	0.1582	12.078	12.0784	17.6359	0.3298
3.5	12.0815	12.0815	8.2955	0.2376	13.8081	13.805	12.1556	0.2519	16.7356	16.7335	17.2452	0.3831
4	24.4118	24.4105	13.6052	0.4696	29.2531	29.253	27.8375	0.5809	29.6063	29.6089	15.2094	0.6273



(a) With landmarks on circle 1



(b) With landmarks on circle 2



(c) With landmarks on circle 3

Figure 3.7: Standard deviations

3.3 Conclusion

This chapter introduces a novel effective localization algorithm for mobile robots based on one single image of a few identified landmarks taken by an onboard camera. The visual angle between two landmarks can be derived from their projections in the same image. The distances between the optical center and the landmarks can be calculated from the visual angles and the known landmark positions based on the law of cosine. The robot position can then be determined using the principle of trilateration. Finally, the robot orientation is computed from the robot position, landmark positions and their projections.

In order to confirm effectiveness and verify accuracy of the localization algorithm, an experiment is designed and conducted with our BlueFox camera equipped at a Pioneer 3 DX robot. The results show algorithm errors under different circumstances. Overall speaking, the closer the robot is to the landmarks, the smaller the errors are. However, we didn't see very different accuracies when using landmarks on different circles. Moreover, using an extra landmarks (4 landmarks) also doesn't lead to obvious improvement of accuracy, most likely the errors caused by noise is too big and hide the accuracy improvement. And at last, generally, the overall accuracy of the trilateration algorithm is at a very good level and can be used for real indoor robot navigation.

Chapter 4 Multi-robot Deployment, Collision Avoidance and Experimental Study on Pioneer 3 DX

This chapter focuses on a decentralized self-deployment scheme for a team of nonholonomic mobile robots forming sensor coverage in a targeted environment while maintaining communication connections and avoiding collisions. A study of our algorithm and another potential field algorithm incorporated in our uniform framework is made. Collision avoidance is discussed and an effective collision avoidance scheme is proposed.

A multi-robot system consists of a collection of networked mobile robots collaborating in task execution. Multi-robot systems have numerous applications, such as surveillance, environment monitoring, disaster rescue, deep sea and planetary exploration, etc. Many of these applications require deploying a team of mobile robots, which are equipped with sensors and/or manipulators, into an environment to provide desired sensory coverage and perform collaborative tasks while maintaining communication connections.

The rest of this chapter is organized as follows. To make the chapter self-contained, Section 4.1 will review our decentralized control framework for deploying multiple nonholonomic mobile robots. Section 4.2 will introduce a control law for accomplishing desired sensor coverage while maintaining communication connections, simulations are conducted to verify effectiveness of our method. In order to verify generality of the control framework, another potential field algorithm is incorporated and results are listed. Section 4.3 will discuss collision in deployment process and a new collision avoidance scheme is proposed to solve this problem. Simulation results show effectiveness of our method. Section 4.5 will summarize this work.

4.1 Control Framework

The work of this chapter focuses on the situation that a team of nonholonomic mobile robots are deployed into a general 2D environment, from a compact initial system gathering, to form desired sensor coverage while maintaining necessary communication connections. Adopting a simplified model for mobile robots in a 2D space, we assume that the state of a mobile robot R_i is defined by its position coordinates (x_i,y_i), its orientation angle θ_i and their time derivatives in a global frame of reference defined in this space.

For a team of N nonholonomic wheeled robots to deploy into a targeted 2D environment to establish sensor coverage while maintaining communication connections, the desired global configuration of the whole multi-robot system will be at first broken down into a collection of desired local configurations, each of which is defined in the neighborhood of an individual robot R_i. Then R_i will self-deploy towards the desired local configuration by moving in the direction of reducing the difference between the desired and instantaneous local configurations, based on the detection of the state of its neighborhood.

An optimal self-deployment motion of R_i can be defined according to Hamilton's principle in classical mechanics [42]. To do that, we at first define an extended Lagrangian for R_i as

$$L_i^E = T_i - V_i - U_i \tag{4.1}$$

where T_i denotes the kinetic energy, V_i denotes the actual potential energy, and U_i denotes an artificial potential energy which moves R_i towards the desired the local configuration. In general, U_i is defined according to the difference between the desired and instantaneous local configurations, based on the online sensory and communication feedback of the state of R_i neighborhood.

Then, based on the Hamilton's principle [42], the optimal motion of R_i should minimize the action of R_i during the deployment process, i.e.

$$[x_i(t), y_i(t), \theta_i(t)] = \arg \min_{x_i(t), y_i(t), \theta_i(t)} \int_{t_1}^{t_2} L_i^E dt \tag{4.2}$$

Meanwhile, due its nonholonomic nature, R_i must satisfy the nonholonomic constraint during its movement

$$\dot{x}_i \sin\theta_i - \dot{y}_i \cos\theta_i = 0 \tag{4.3}$$

which means that it can only have a non-zero speed in its longitudinal direction (i.e. along its orientation) while its side speed is always zero. Moreover, to guarantee the convergence of R_i towards its desired local configuration, a virtual Rayleigh's dissipation function is adopted to provide the necessary damping

$$F_i = \frac{1}{2}(k_{ix}\dot{x}_i^2 + k_{iy}\dot{y}_i^2 + k_{i\theta}\dot{\theta}_i^2) \tag{4.4}$$

where k_{ix}, k_{iy} and $k_{i\theta}$ are the damping coefficients associated with the linear and angular velocities of R_i respectively.

Based on the above considerations and using the technique of variational calculus [42], we can derive the following Lagrange's equation for R_i

$$\begin{aligned}
&\frac{d}{dt}\left(\frac{\partial(T_i - V_i)}{\partial \dot{x}_i}\right) - \frac{\partial(T_i - V_i)}{\partial x_i} + \frac{\partial F_i}{\partial \dot{x}_i} - \lambda_i \sin\theta_i = f_{ix} \\
&\frac{d}{dt}\left(\frac{\partial(T_i - V_i)}{\partial \dot{y}_i}\right) - \frac{\partial(T_i - V_i)}{\partial y_i} + \frac{\partial F_i}{\partial \dot{y}_i} + \lambda_i \cos\theta_i = f_{iy} \\
&\frac{d}{dt}\left(\frac{\partial(T_i - V_i)}{\partial \dot{\theta}_i}\right) - \frac{\partial(T_i - V_i)}{\partial \theta_i} + \frac{\partial F_i}{\partial \dot{\theta}_i} = f_{i,\theta} \\
&\dot{x}_i \sin\theta_i - \dot{y}_i \cos\theta_i = 0
\end{aligned} \tag{4.5}$$

where

$$\begin{aligned}
f_{ix} &= \frac{d}{dt}\left(\frac{\partial U_i}{\partial \dot{x}_i}\right) - \frac{\partial U_i}{\partial x_i} \\
f_{iy} &= \frac{d}{dt}\left(\frac{\partial U_i}{\partial \dot{y}_i}\right) - \frac{\partial U_i}{\partial y_i} \\
f_{i\theta} &= \frac{d}{dt}\left(\frac{\partial U_i}{\partial \dot{\theta}_i}\right) - \frac{\partial U_i}{\partial \theta_i}
\end{aligned} \tag{4.6}$$

are the generalized forces associated with x, y and θ, which define the control law for the self-deployment motion of R_i. At each time t, R_i can online calculate its desired acceleration for the deployment motion from (5) based on the state of its neighborhood.

Equation (4.5) provides a general framework for controlling the self-deployment motion of individual nonholonomic mobile robots during the multi-robot deployment process. In practice, the specific control law needs to be defined according to the coverage requirement of the specific multi-robot deployment task. By moving each mobile robot in the way defined by its equation of self-deployment motion, eventually the resulting local coverage in the neighborhoods of all the robots altogether will form a global multi-robot coverage to the targeted environment.

4.2 Study of Two Kinds of Potential Field Forces

4.2.1 Control Laws

In order to form the sensor coverage over a target environment, a desired configuration of the multi-robot system can be that each mobile robot reaches desired distances with neighboring robots.

Considering the limitations in robot sensing and communication, we define a desired distance a_{ij} between a robot R_i and a neighboring robot R_j as a designated distance over which an enough far and reliable in-between sensor coverage can be established and a reliable in-between wireless communication connection can be maintained. In reality, those robots have limited sensing ranges, and, due to noise and loss of resolution, the sensing performance has some unreliability and usually degrades with the distance [45-53]. It is more realistic to define a confident sensing range for R_i based on sensor calibration such that R_i has a sufficient sensing confidence within this range. Meanwhile, in order to maintain communication connections with nearby robots, Ri must ensure that the signals from neighboring robots are received with sufficient signal to noise ratio such that a signal transmitted by a nearby robot R_j can be successfully received and decoded by R_i. The determination of a reliable communication range is far from trivial, because it is easily affected by various environmental factors in real time, such as surrounding objects and atmospheric conditions. However, a communication calibration and a conservative estimation are helpful. Moreover, because the communication range is usually much larger than a confident sensing range, the a_{ij} determined based on the sensing confidence is usually sufficient for both the sensor coverage and communication maintenance purposes.

In order to approach and maintain the desired distance, the control law should be defined in such a way that, when the distance d_{ij} between R_i and R_j is smaller than a_{ij}, an repulsive force should be applied to push the robots apart from each other, and the force should increase as the distance decreases; when the distance between R_i and R_j is greater than aij, an attractive force should be applied to pull the robots towards each other, and the force should increase as the distance increases. Based on this consideration, we can define the artificial force acting on R_i by all R_j in its neighborhood as:

$$\begin{aligned} f_{ix} &= \sum_j w_{ij} k_i sign(a_{ij} - d_{ij}) \mid a_{ij} - d_{ij} \mid^c \frac{x_{ij}}{d_{ij}} \\ f_{iy} &= \sum_j w_{ij} k_i sign(a_{ij} - d_{ij}) \mid a_{ij} - d_{ij} \mid^c \frac{y_{ij}}{d_{ij}} \\ f_{i\theta} &= \chi_i (\text{atan2}(\sum_j w_{ij} x_j - x_i, \sum_j w_{ij} y_j - y_i) - \theta_i) \end{aligned} \tag{4.7}$$

where $x_{ij} = x_j - x_i$, $y_{ij} = y_j - y_i$, $d_{ij} = \sqrt{x_{ij}^2 + y_{ij}^2}$, c is a positive real exponent, sign(.) denotes the sign of the enclosed expression, k_i denotes the force coefficient for R_i, χ_i denotes the torque coefficient for R_i, and w_{ij} denotes a coefficient weighting the effect of R_j on R_i with $\sum_{\text{j}} w_{ij} = 1$.

Here, $f_{i\theta}$ defines an artificial torque to steer R_i towards the center of its neighborhood, which helps to balance the local configuration.

Besides the power function force law in (4.7), inverse-power force law is also widely adopted [24,26]. In the context of approaching and maintaining the desired distance, the force acting on R_i by all R_j in its neighborhood can be defined as:

$$\begin{aligned} h_{ix} &= \sum_j w_{ij} \left(\frac{k_{ri}}{d_{ij}^{c_r}} - \frac{k_{ai} sign(b_{ij} - d_{ij})}{\mid b_{ij} - d_{ij} \mid^{c_a}} \right) \frac{x_{ij}}{d_{ij}} \\ h_{iy} &= \sum_j w_{ij} \left(\frac{k_{ri}}{d_{ij}^{c_r}} - \frac{k_{ai} sign(b_{ij} - d_{ij})}{\mid b_{ij} - d_{ij} \mid^{c_a}} \right) \frac{y_{ij}}{d_{ij}} \\ h_{i\theta} &= \chi_i (\text{atan2}(\sum_{\text{j}} w_{ij} x_j - x_i, \sum_{\text{j}} w_{ij} y_j - y_i) - \theta_i) \end{aligned} \tag{4.8}$$

where $\frac{k_{ri}}{d_{ij}^{c_r}}$ defines an repulsive force added on R_i by R_j, which makes the robots cover as much area as possible, and $\frac{k_{ai} sign(b_{ij} - d_{ij})}{|b_{ij} - d_{ij}|^{c_a}}$ defines an attractive force added on R_i by R_j when $d_{ij} < b_{ij}$, which suppresses the coverage gap between the robots to be less than a prohibitive distance b_{ij}. Here, c_r and c_a are positive real exponents, while k_{ri} and k_{ai} are repulsive and attractive force coefficients for R_i respectively. In particular, when c_r=c_a and k_{rij}=k_{aij}, we can set b_{ij}=$2a_{ij}$, because then h_{ix}= h_{iy}=0 when the distance d_{ij} between R_i and R_j reaches the desired distance a_{ij}.

4.2.2 Initial Robots Generation

Initial coordinates and orientations of robots play a significant role in deployment process and greatly affect eventual results. In order to better evaluate the algorithms, these coordinates and orientations have to be random. Besides, considering that the robot's shape, distance between centers of robots should be at least bigger than diameter of the robot. The generation of the initial positions and orientations is described as following.

Firstly, a pair of random coordinates $x_i \in [x_{start}, x_{end}]$ and $y_i \in [y_{start}, y_{end}]$ with a random orientation $\theta_i \in [-\pi, \pi]$ are assigned to the first robot, then another set of random coordinates and orientation are generated in the same range and compared with the first one. To make sure they are not overlapped, Euclidean distance between these two pairs of coordinates should be bigger than 2r (r is radius of robots). Otherwise, this set of numbers is discarded and another set of random numbers will be generated until qualified numbers are obtained. Coordinates and orientations of all the remaining 48 robots are obtained in the same way.

Apparently, initial deployment area should be big enough for robots to be located properly by using our generating method. The area should be at least but not limited to number of robots (e.g., 50) times single robot's physical coverage area (πr^2). For example, if robot's radius is 0.3m and we have 50 robots, then the threshold is $50 \times \pi \times 0.3^2 = 14.1372 m^2$. Of course in order to get this deployment done quickly, it is usually necessary to set a much bigger area. (How large the area is required for a quick deploy is a topic that needs further discussion and beyond scope of this chapter).

In this chapter, 4 different area sizes will be tested for algorithm verification, 5m×5m, 6m×6m, 7m×7m and 8m×8m.

4.2.3 Neighborhood Definition

Definition of neighborhood is also critical to algorithm performance, we will go through the simulations using two different neighborhood definitions.

The first definition of neighborhood is called traditional definition. It is commonly accepted that, for a robot R_i, robots that are within its sensing or communication range are considered its neighbors. This is our first definition of neighborhood, which is a circle with R_i in its center.

However, this definition sometimes results in too many neighbors, especially in initial stage of deployment when all robots stay very close. This situation not only causes delays in communications and decision making, but greatly increases risk of unexpected robot behaviors due to combined effect of interactive forces. Alternatively, a topological neighborhood of R_i is defined according to a topological graph representation of the multi-robot system [32,45], which limits the neighborhood of R_i to its one-hop neighbors on the graph, and this is our second type of neighborhood.

4.2.4 Other Physical Constraints

In practice, a robot always has velocity and acceleration limits. In other words, a robot is subject to its moving capability. This constraint is taken into account in our situation

$$v_a = \begin{cases} v_{\max}, & \text{if } v_c > v_{\max} \\ -v_{\max}, & \text{if } v_c < -v_{\max} \\ v_c & \end{cases} \tag{4.9}$$

$$a_a = \begin{cases} a_{\max}, & \text{if a}_c > a_{\max} \\ -a_{\max}, & \text{if a}_c < -a_{\max} \\ a_c & \end{cases} \tag{4.10}$$

where v_{max} is robot's maximum velocity (either linear or angular) and a_{max} is its maximum acceleration (either linear or angular).

In our algorithm, component accelerations in x and y directions are calculated separately, so when it is detected that resulted acceleration is bigger than the limit, it is necessary to derive each direction's acceleration limits accordingly.

As we know, nonholonomic robots are subject to (4.3), from which we can derive that:

$$\ddot{x}\sin\theta + \dot{x}\cos\theta\cdot\dot{\theta} - \ddot{y}\cos\theta + \dot{y}\sin\theta\cdot\dot{\theta} = 0 \tag{4.11}$$

Combing with:

$$\sqrt{\ddot{x}^2 + \ddot{y}^2} = a_{\max} \tag{4.12}$$

We get following:

$$\ddot{y} = -(\dot{y}\sin\theta - \dot{x}\cos\theta)\cos\theta \pm \sin\theta\sqrt{k - (\dot{y}\sin\theta - \dot{x}\cos\theta)^2} \tag{4.13}$$

$$\ddot{x} = \frac{\ddot{y}\cos\theta + \dot{y}\sin\theta - \dot{x}\cos\theta}{\sin\theta} \tag{4.14}$$

This is the real accelerations in x and y directions if calculated acceleration is bigger than acceleration limit.

4.2.5 Simulation Results

In this simulation, number of critical connections for Control Law I is k=6, radius of robot r=0.3m, initial deployment area size is 7m×7m, $k_{degree}=k_{cover}=1$, mass of robot is 12kg, R_c=6m (to make final distances between robots equal to 3 (when $f_{degree}+f_{cover}=0$)), time interval between each step is 0.2s, maximum linear and angular accelerations are 0.2m/s^2 and 5 degrees/ s^2, maximum linear and angular velocity limits are 0.4m/s and 5 degree/s, damping coefficients $k_{ix} = k_{iy} = 5$, $k_{i\theta} = 1$. The complete results are listed as following, and final configuration of robots, average distance and number of collisions.

Figure 4.1 and Table 4.1 show results of Control Law I and II's performance. In this example, order of both algorithms is 1. In Figure 4.1, the first and second row show algorithms' performance under traditional neighborhood definition, which considers all robots within communication range as its neighbors, the third and fourth row show the performance under the modified definition of neighborhood, which is generated using Gabriel graph. The left column shows final configurations and right column shows average distances at different steps. As we can see that the performance has been improved significantly either in average distance convergence or final configuration. Table 4.1 shows number of collisions under different neighborhood definitions with different control laws, and it also confirms the conclusion.

In the tables, "N/A" means collisions still exist after the given period.

Figure 4.2 and Table 4.2 show results of the two different control laws with second order definition. Figure 4.3 and Table 4.3 show results of the two different control laws with third order definition. The figures illustrate that Gabriel neighborhood definition is superior to traditional neighborhood definition which is based on communication range. Gabriel neighborhood is better at picking the most important neighbors and helping potential field algorithm to better deploy robots, making the deployment more organized and even. Besides, order of the algorithm also partly decides the final results. From our simulation, algorithm with order 2 is the best since it has the best final deployment and collisions are the least.

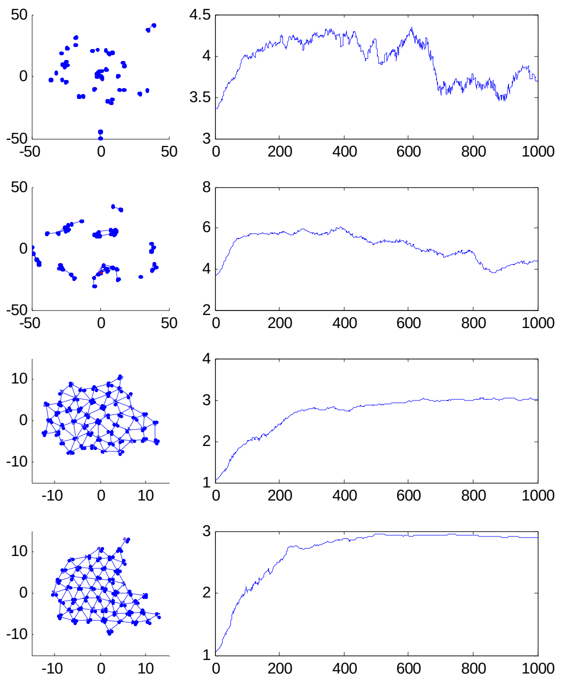


Figure 4.1: Final deployment and average distances,

traditional neighborhood VS. Gabriel neighborhood, n=1

Table 4.1: Number of collisions when n=1

Control Law	Neighborhoo d	Order	Number of maximum collisions	Collisions disappear at step:
I	Traditional	1	21	N/A
II	Traditional	1	18	N/A
I	Gabriel	1	4	190
II	Gabriel	1	12	171

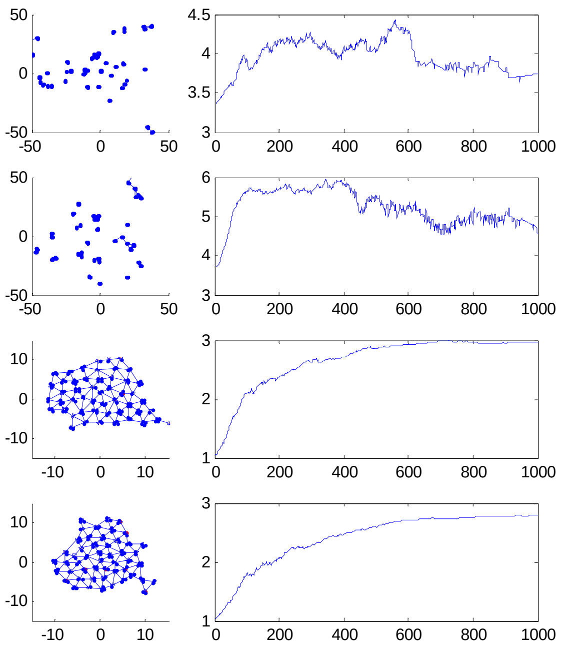


Figure 4.2: Final deployment and average distances,

traditional neighborhood VS. Gabriel neighborhood, n=2

Table 4.2: Number of collisions when n=2

Control Law	Neighborhoo d	Order	Number of maximum collisions	Collisions disappear at step:
I	Traditional	2	17	834
II	Traditional	2	20	569
I	Gabriel	2	2	29
II	Gabriel	2	7	242

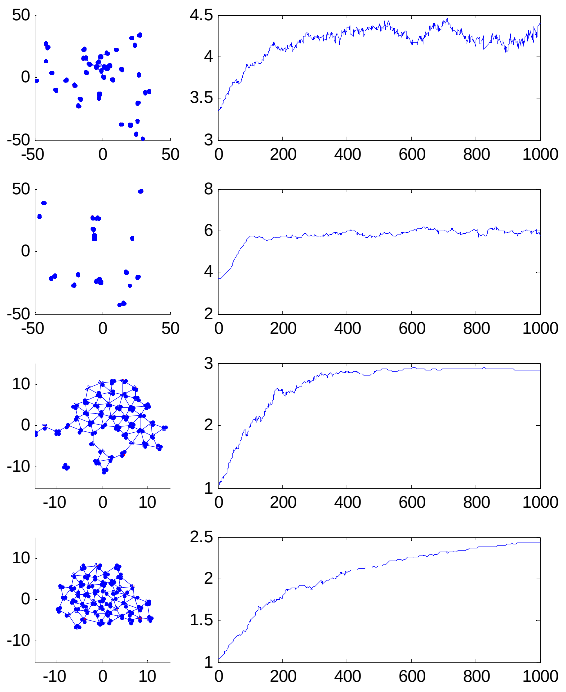


Figure 4.3: Final deployment and average distances,

traditional neighborhood VS. Gabriel neighborhood, n=3

Table 4.3: Number of collisions when n=3

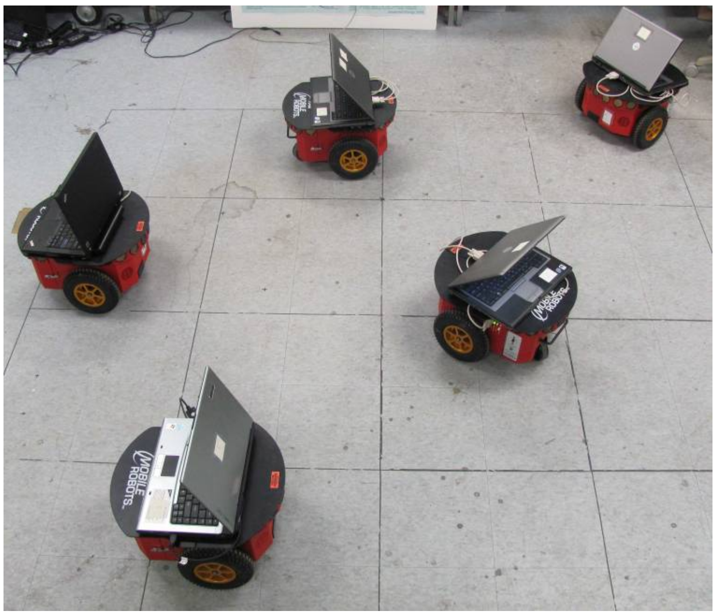
Control Law	Neighborhoo d	Order	Number of maximum collisions	Collisions disappear at step:
I	Traditional	3	13	895
II	Traditional	3	26	887
I	Gabriel	3	9	N/A
II	Gabriel	3	4	223

4.2.6 Real Experiment

In this section, a group of 5 well calibrated Pioneer 3 DX robots is used to verify performance of our algorithm. The experiment takes most settings used in simulations, including velocity and acceleration limit, mass of robot, radius of robot. The desired distance between robots is 1.2 meters. Our own algorithm is implemented in C++ and the order of the algorithm is 2, as we picked in the last section.

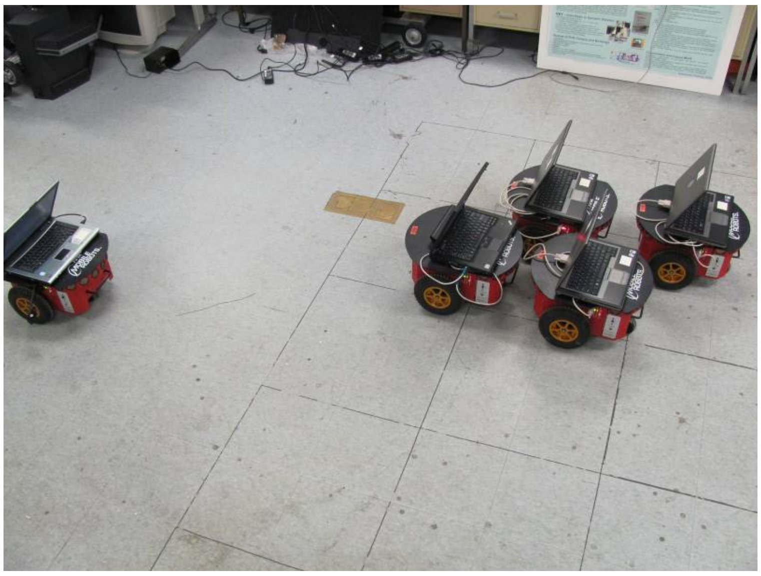
Three experiments are conducted in this section. The 5 Pioneer 3 DX robots are given three different sets of initial positions and orientations respectively. As we can see from the results, the robots deploy to the environment smoothly. In all three experiments, after around 20 seconds, distances among robots are very close to our setting desired distance and no collision is observed.

(a) Initial distribution

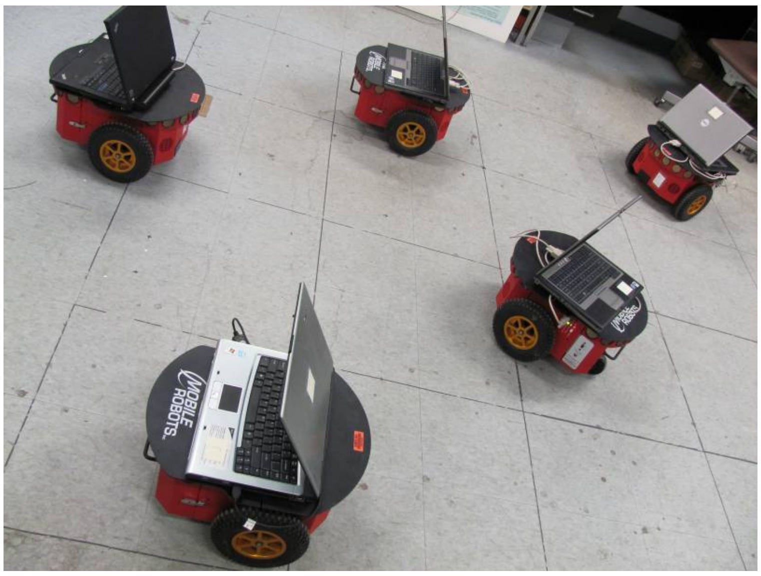


(b) Distribution after 40s

Figure 4.4: Experiment 1

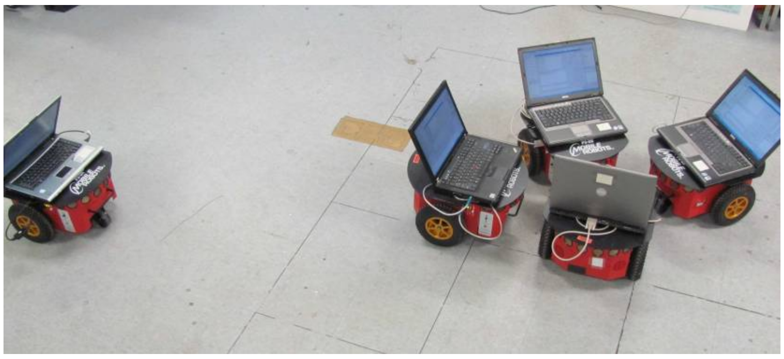


(a) Initial distribution

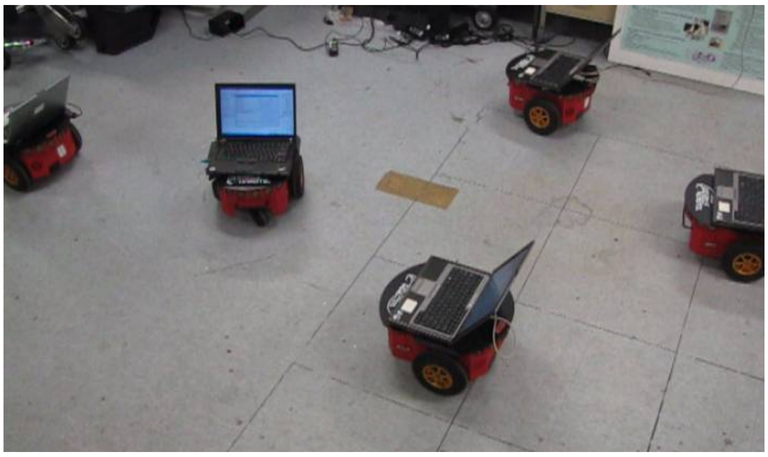


(b) Distribution after 40 seconds

Figure 4.5: Experiment 2



(a) Initial distribution



(b) Distribution after 40s

Figure 4.6: Experiment 3

4.2.7 Conclusion

Targeting robust deployment of multiple mobile robots under realistic constraints, this section has proposed a decentralized control scheme for reliably accomplishing desired sensor coverage among a team of nonholonomic mobile robots while maintaining necessary communication connections. The results show that robots are successfully deployed into an open space in an even order if certain parameters are appropriately picked, this applies to both methods- our method and Poduri's method.

The collisions during deployment are also given concern and numbers of collisions are collected in various set of simulations. The results inspired us to think about a way to avoid collisions. Actually, collisions are more likely to happen with large and dense initial population. For example, in our real experiment with 5 Pioneer 3 DX robots, no collision happened.

The real experiments also prove that our algorithm without consideration of collision avoidance can be used in teams with fewer numbers of members.

However, in order to completely solve deployment problem, collision avoidance is an inevitable step. Our next step is to propose a guide of designing collision avoidance schemes that are able to avoid collisions under any given circumstance.

4.3 Collision Avoidance Scheme I – Slowing Down

Collision avoidance is a fundamental problem in multi-robot deployment. Most of existing potential/force field methods rely on repulsive forces among robots and obstacles to avoid collisions. However, this scheme has some limitations. First, the control force on R_i combines the effects of all the other robots in R_i neighborhood. The combined force, due to the complex interactions between R_i and its neighbors, may still attracts R_i towards R_j though R_j individually tries to push R_j away. Second, as a second-order dynamic system, a mobile robot has a delayed response to the changes in the potential/force field. Because the robot velocity is the time integration of the acceleration control, the effect of the acceleration control on collision avoidance will take place with some delay. Thus, it is difficult for two nearby robots, which approach each other very fast, to avoid collisions. It is even more difficult to handle when more robots are in the risk of collision. Third, when robots are very close to each other, the repulsive forces can be huge, which often result in fast and aggressive movements of robots and make it difficult to maintain the desired task-specific correlation among the robots. Collision avoidance becomes even more challenging for nonholonomic mobile robots, due to the constraint on their instantaneous motion directions. Moreover, the constraints on maximal robot velocity and acceleration add some uncertainty to collision avoidance control.

We take a convenient and effective solution to the collision avoidance problem in the deployment of multiple nonholonomic mobile robots by letting a mobile robot slow down quickly once it detects risk of collision. We set an alerting inter-robot distance. When a robot detects other robots within this distance by either sensing or communications, it will check if they are indeed moving closer. If so, the robot will trigger a collision avoidance control law; if not, the robot continues with its normal deployment process. Unlike other motion or position based collision avoidance schemes, our method fits into the acceleration control framework discussed in previous sections, and maintains the smoothness of robot movement, which is more realistic to physical robot systems. Though the deployment process may slow down locally and temporarily, it generally results in a smooth deployment process and a better equilibrium coverage configuration.

Apparently, it is unnecessary to do collision avoidance if two robots are moving away from each other, even though they are very close in distance. Therefore, the collision avoidance control does not need to be turned on all the time, in order to maximize the efficiency of deployment. This decision can be made based on the calculation of the relative position and movement between the two robots. From Figure 4.7, we can see that, for two robots R_i and R_j, we can determine if R_i and R_j are approaching each other based on the angle α between $\mathbf{p}_{ij}$ and $\mathbf{v}_{ij}$

$$\alpha = \arccos\left(\frac{\mathbf{v}_{ij} \cdot \mathbf{p}_{ij}}{|\mathbf{v}_{ij}||\mathbf{p}_{ij}|}\right), \tag{4.18}$$

where "·" denotes the dot product, and "| |" denotes the norm. Here $\mathbf{p}_{ij}$ is the position of R_j relative to R_i, and $\mathbf{v}_{ij}$ is the velocity of R_i relative to R_j. In particular, R_i and R_j are getting closer to each other only if $\alpha>90°$ (Figure 4.7a); they are passing by each other when $\alpha=90°$ (Figure 4.7b); they are moving apart from each other if $\alpha<90°$ (Figure 4.7c).

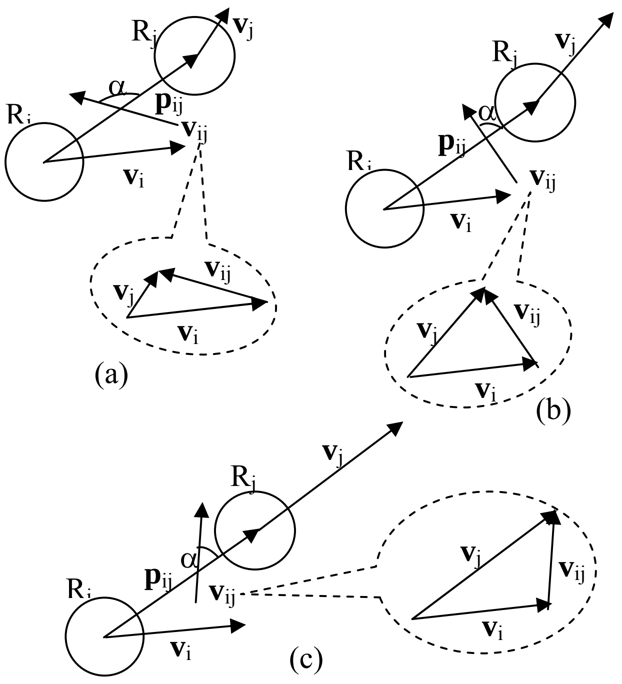


Figure 4.7: Collision detection

Upon the confirmation of the risk of collision, i.e. at least another robot is within the alerting distance and approaching R_i, a resistive control force will be added to the deployment control input of R_i as

$$\begin{aligned} f_{rix} &= -k_{ci}\dot{x}_i \\ f_{riy} &= -k_{ci}\dot{y}_i, \end{aligned} \tag{4.19}$$

where k_{ci} is the force coefficient for R_i. Comparing with (4.5), this is equivalent to increasing the damping coefficients of the dissipation function, which slows down R_i. Moreover, because of the symmetry in collision detection, i.e. if R_i detects the risk of colliding with R_j, R_j should also detect the risk of colliding with R_i, the robots with the risk of collisions will all slow down. Consequently the potential collisions will be suppressed.

By locally slowing down the robots with collision risks, the proposed collision avoidance scheme allows those robots without collision risks to spread out first. Consequently the local robot density will be reduced and the risk of collisions will then diminish. Once the risk of collision disappears, R_i will resume its normal self-deployment process by removing the resistive control force (4.19). In this way, the proposed scheme helps robots avoid collisions while maintaining the deployment progress.

4.3.1 Results for Holonomic Robots

Results for holonomic robots will provide us evidence how our collision avoidance schemes works on different kinds of robots. Generally, holonomic robots should perform better than nonholonomic robots in spreading out since it has ability to move at any direction as desired. In contrast, nonholonomic's movements are subject to nonholonomic constraints, and this restrains its ability to respond quickly to risk of collisions.

To better evaluate performance of our scheme, results before and after collision avoidance schemes applied are listed in the same figure.

50 holonomic mobile robots are used in the simulations. Each robot has an approximated circular footprint with a radius of 0.3m and a mass of 12kg. For deployment control, both definitions of the control force in (4.7) and (4.8) were tested. The involved common coefficients were set as $c=c_a=c_r=2$, $k_i=k_{ri}= k_{ai}=1$, $k_{ci}=1.2$, $k_i=2$, $w_{ij}=1$ and $i=1$. Moreover, when (4.7) is applied, $k_{ix}=k_{iy}=0.8$; when (4.8) is applied, $k_{ix}=k_{iy}=1.2$. In addition, each robot allows a maximal linear acceleration of $0.2m/s^2$, a maximal angular acceleration of $5degree/s^2$, a maximal linear speed of 0.4m/s, and a maximal angular speed of 5degree/s. Initially, the robots are uniformly distributed in a 3.5m×3.5m area with $0.25m \leq d_{ij} \leq 0.35m$ among neighboring robots. For sensor

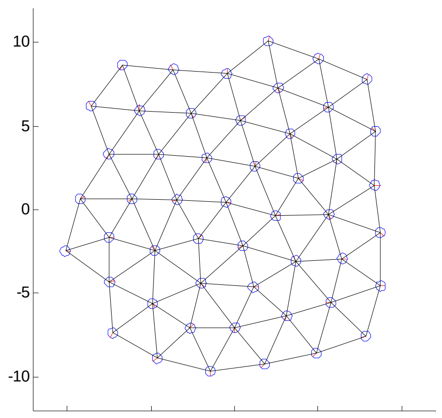
coverage, each robot intends to approach a desired distance of 3m with neighboring robots. However, it will switch to the collision avoidance mode when its distance with any neighboring robot gets below the alerting distance of 0.9m. We set the time interval for state update and decision making as 0.2s. Totally 12 trials with different random initial distributions of the robots were made for each of the control forces (4.7) and (4.8) with and without applying the collision avoidance strategy. For each trial, 2000 steps of system evolution were recorded.

Figure 4.8 and Figure 4.9 show the typical deployment outcomes of the proposed scheme using the control forces defined in (4.7) and (4.8) respectively but without taking the proposed collision avoidance strategy. It shows that both definitions of the deployment control law lead to a satisfactory team coverage configuration (typical configurations are displayed in Figure 4.8a and Figure 4.9a), and, in particular, the inter-robot distance converges to the desired coverage distance (average curves over the trials are displayed in Figure 4.8b and Figure 4.9b).

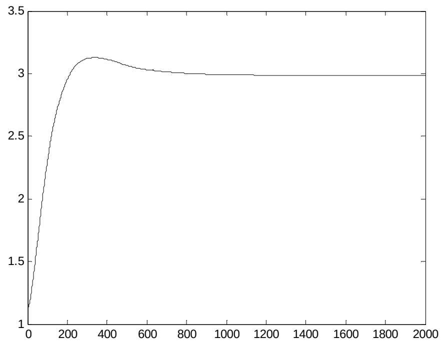
However, neither of the deployment control laws completely eliminated the collisions. In the simulations, with control law (4.7), up to 4 collisions were observed at some moments; control law (4.8) turns out better in collision avoidance, but 1 to 2 collisions were consistently observed across the trials. These collisions mostly happened at the early stage of the deployment process when the involved robots were densely gathered, within the first 300 steps for control law (4.7) and within the first 200 steps for control law (4.8).

Further simulations showed that, by applying the proposed collision avoidance strategy, collisions were completely avoided across the trials. Figure 4.10 and Figure 4.11 show the typical deployment outcomes of the proposed scheme using the control forces defined in (4.7) and (4.8) respectively and taking the proposed collision avoidance strategy. It shows that the combination of deployment control and collision avoidance control leads to a satisfactory team coverage configuration (as displayed in Figure 4.10 and Figure 4.11) and convergence of inter-robot distance (as displayed in Figure 4.10b and Figure 4.11b) while avoiding collisions.

In particular, the convergence rates of the average inter-neighbor distance under different control laws before/after applying the collision avoidance strategy were recorded, and a set of average data across different trials is presented in Table 4.4. Here we define the rate of convergence as the number of steps for the robot team to settle within 5% of the equilibrium average inter-neighbor distance (i.e. settling time). The data show the compromise in the rate of convergence as the result of applying the collision avoidance strategy.

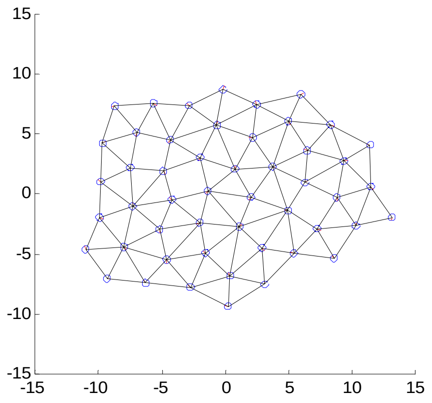


(a) Equilibrium team

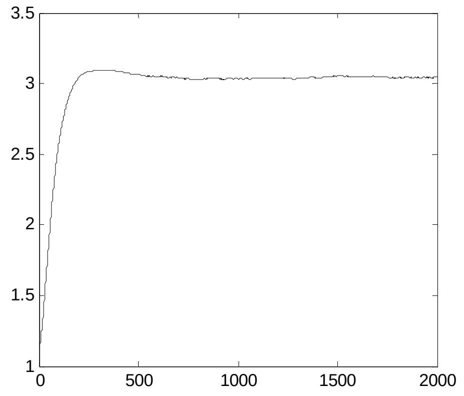


(b) Convergence of average inter-neighbor distance

Figure 4.8: Deployment simulation with control law (4.7) only

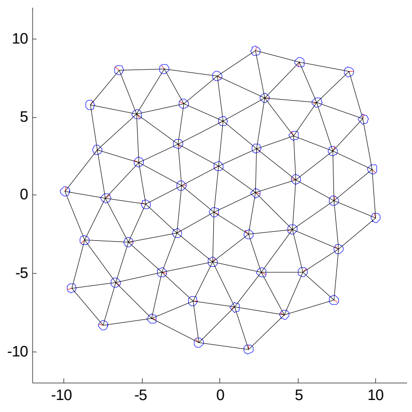


(b) Equilibrium team

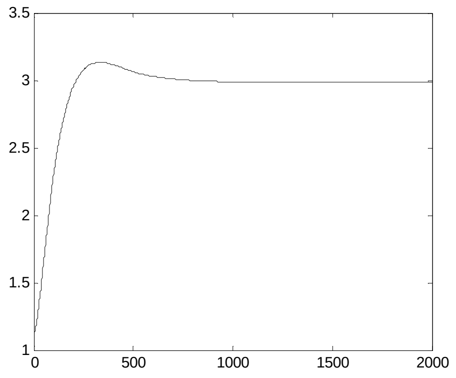


(b) Convergence of average inter-neighbor distance

Figure 4.9: Deployment simulation with control law (4.8) only

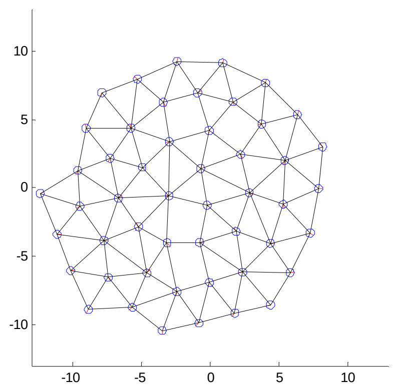


(a) Equilibrium team

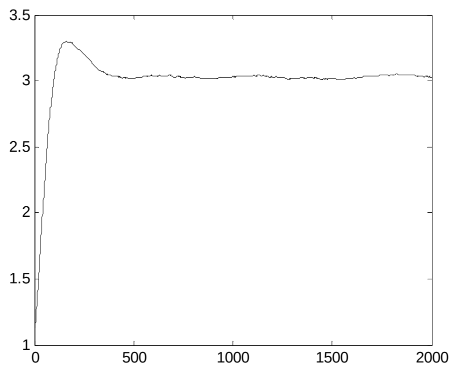


(b) Convergence of average inter-neighbor distance

Figure 4.10: Deployment simulation with control law (4.7) and (4.19)



(a) Equilibrium team



(b) Convergence of average inter-neighbor distance

Figure 4.11: Deployment simulation with control law (4.8) and (4.19)

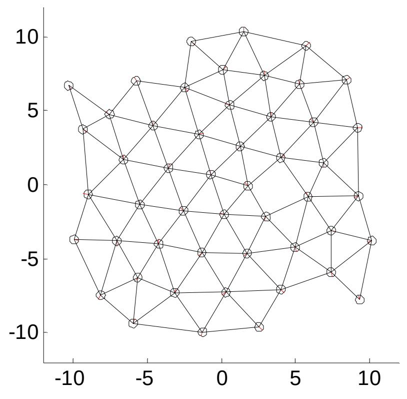
Table 4.4: Rate of convergence

Control law	Collision avoidance strategy applied?	Settling time (steps)
(4.7)	No	171
(4.7)	Yes	356
(4.8)	No	148
(4.8)	Yes	263

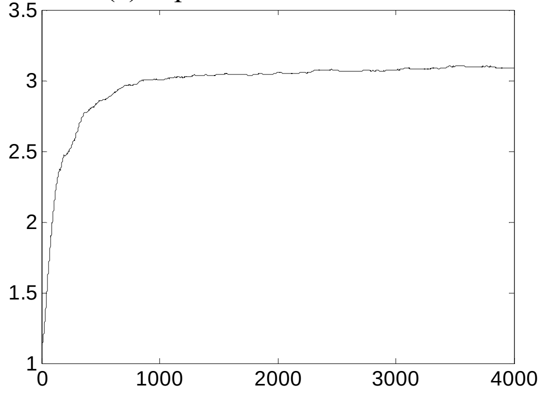
4.3.2 Results for Nonholonomic Robots

For nonholonomic robots simulation, basically all parameters remain the same, excepting that all robots are subject to nonholonomic constraints here. Figure 4.12 and Figure 4.13 show the typical deployment outcomes of the proposed scheme using the control forces defined in (4.7) and (4.8) respectively but without taking the proposed collision avoidance strategy. Figure 4.14 and Figure 4.15 show the deployment outcomes with collision avoidance strategy applied.

The results are very similar to what we got in the previous section. Before taking collision avoidance strategy on both control laws, control law (4.7) has up to 8 collisions over time, control law 4.8 is better but 1 to 2 collisions are observed over the whole deployment process. Collisions are completely removed after collision avoidance strategy is applied and Figure 4.14 and Figure 4.15 show the outcomes. If we use the same definition of convergence rate as used in the previous section, the settling steps are listed in Table 4.5.

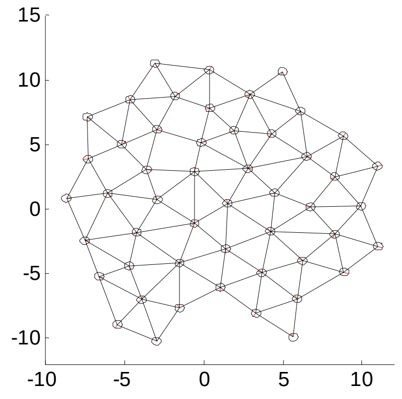


(a) Equilibrium team

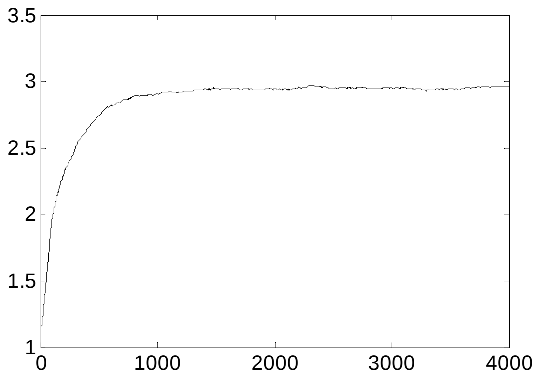


(b) Convergence of average inter-neighbor distance

Figure 4.12: Deployment simulation with control law (4.7) only

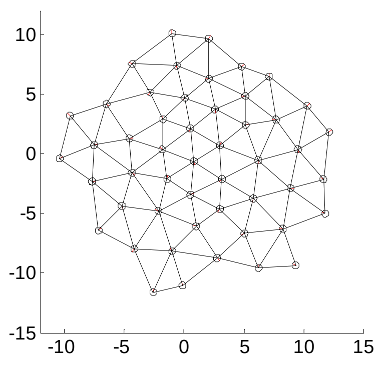


(a) Equilibrium team configuration

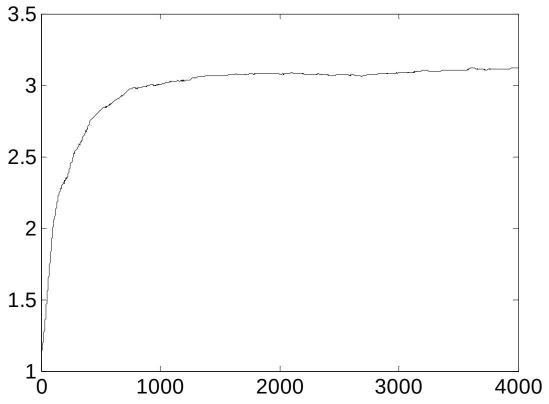


(b) Convergence of average inter-neighbor distance

Figure 4.13: Deployment simulation with control law (4.8) only

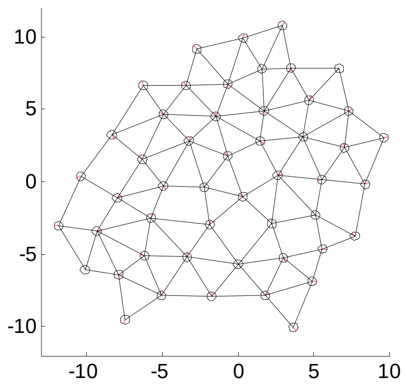


(a) Equilibrium team configuration

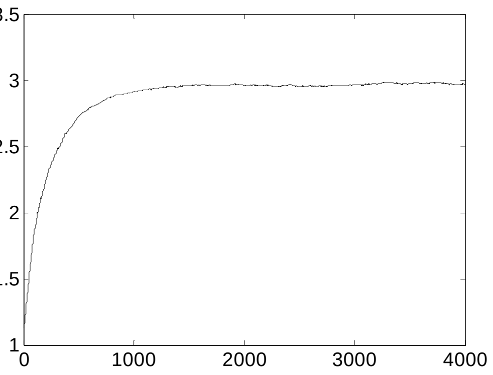


(b) Convergence of average inter-neighbor distance

Figure 4.14: Deployment simulation with control laws (4.7) and (4.19)



(a) Equilibrium team



(b) Convergence of average inter-neighbor distance

Figure 4.15: Deployment simulation with control laws (4.8) and (4.19)

Table 4.5: Rate of convergence

Control law	Collision avoidance strategy applied?	Settling time (steps)
(4.7)	No	655
(4.7)	Yes	672
(4.8)	No	562
(4.8)	Yes	685

4.3.3 Real Experiment for Nonholonomic Robots

Experiments were also conducted on real robots to verify the effectiveness of the proposed deployment scheme under physical operational conditions, in particular with robot state estimation error, communication delay and operational asynchrony among the robots.

The robot team in our experiments consisted of 5 Pioneer 3-DX and 4 Amigobot robots made by Mobile Robots Inc (Figure 4.16). They are differential-drive robots under the nonholonomic kinematic constraint. Each Pioneer 3-DX robot is controlled by an onboard laptop PC, while each Amigobot is controlled by an offboard PC through wireless communications. Each robot is self-localized through odometry (with onboard encoders), and the data of robot states are exchanged among robots (in fact controlling computers) wirelessly through the Wi-Fi (IEEE 802.11) protocol.

Some control-related parameters include the radius of each Pioneer 3-DX 0.3m, mass of each Pioneer 3-DX 12kg (including onboard PC), radius of each Amigobot is 0.1m, mass of each Amigobot is 3kg, and involved coefficients $c=c_a=c_r=2$, $k_{ix}=k_{iy}=1$, $k_{i\theta}=2$, $k_{ij}=k_{rij}=k_{aij}=1$, $k_{ri}=1.2$, $w_{ij}=1$ and $\chi_i=1$. Moreover, we set each robot with a maximal linear acceleration of 0.2m/s^2, a maximal angular acceleration of 5degree/s^2, a maximal linear speed of 0.4m/s, and a maximal angular speed of 5degree/s. Initially, the robots are uniformly distributed in a 3m×3m area, with the center-to-center distance between neighboring robots ranging from 0.7m to 0.9m. We set the desired distance between two neighboring robots as 1.3m. When its distance with any neighboring robot gets below the safety distance of 0.9m, a robot will switch to the collision avoidance mode.

Comparing with simulations, experiments involve more uncertainties. First, motors have some delays in responding to the control input. Second, communications are not in real time, data packets are lost from time to time, and communication delay happens due to constant position and direction change of the robots. Third, odometry-based robot self-localization has accumulative error, even without considering the skidding between wheels and the floor. It is assumed that the first and second factor don't affect overall moving trend of robots. Due to the third factor, each experiment is not allowed to last too long. Our experiment typically lasts 8 minutes and during this period of time we assume the location and orientation information derived from encoders are accurate enough.

Figure 4.16 to Figure 4.19 show the experimental results of deploying the 10-robot team into an indoor environment.

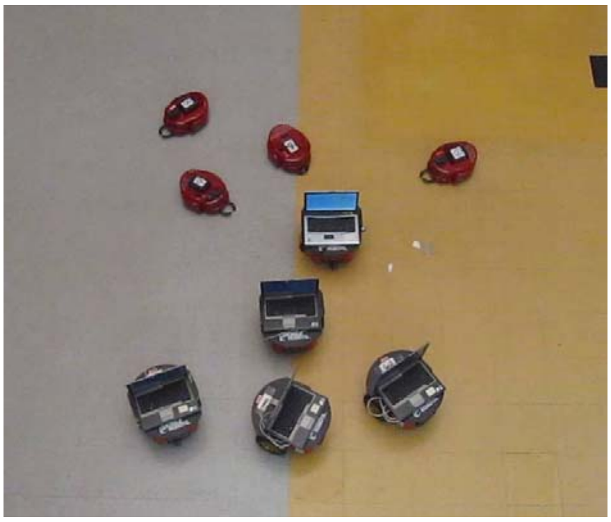


Figure 4.16: Initial deployment of robots

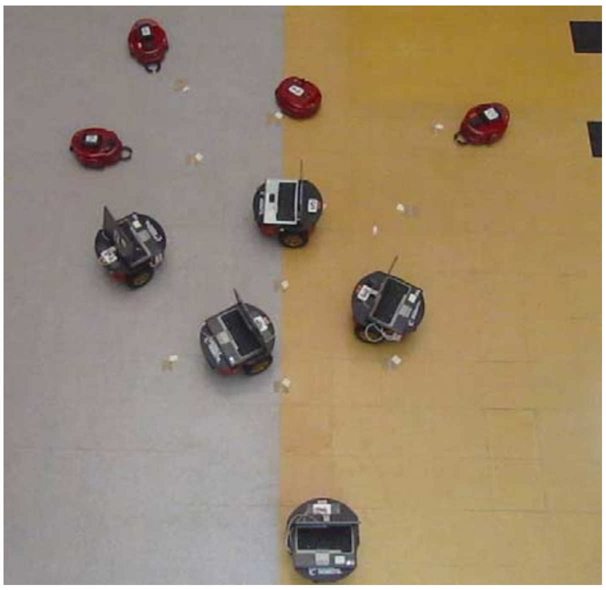


Figure 4.17: After 2 minutes

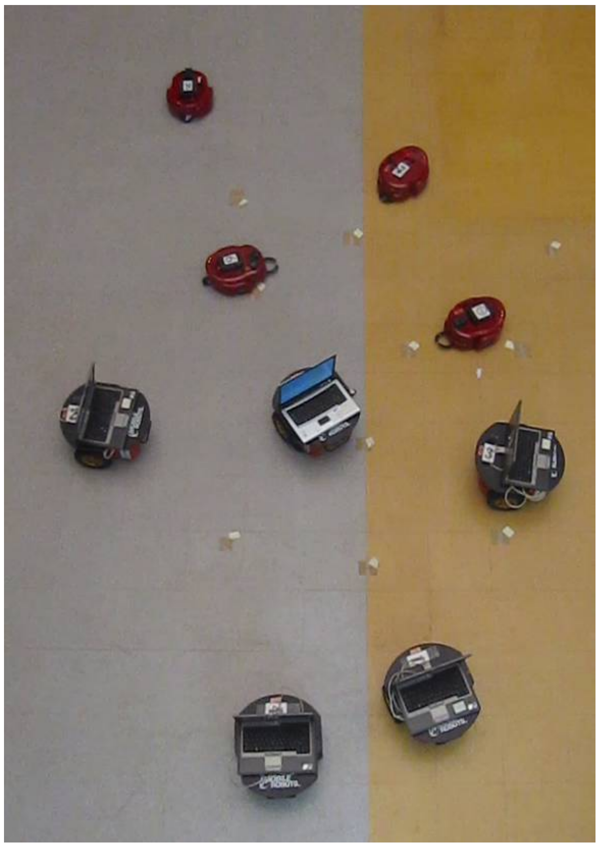


Figure 4.18: After 4 minutes

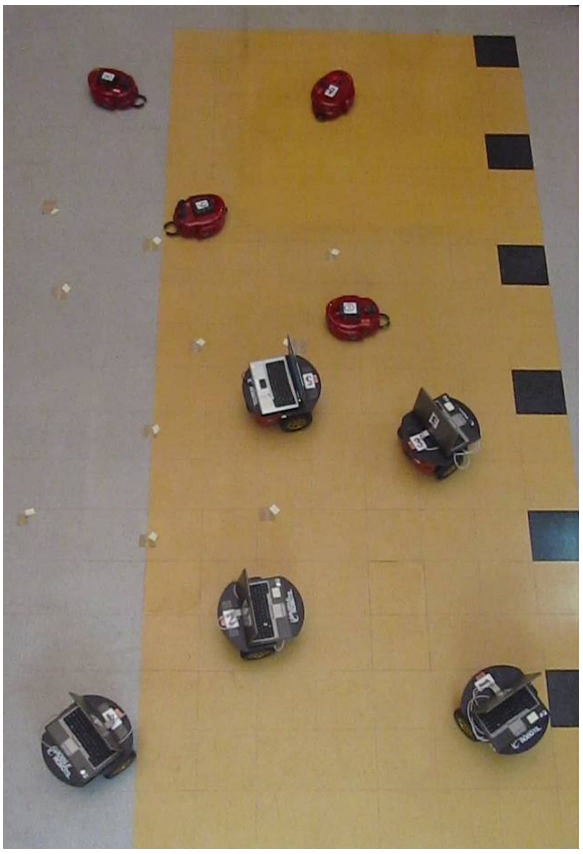


Figure 4.19: After 6 minutes

In order to make the results clearer, robots' positions and orientations are recorded and the data is redrawn. Figure 4.20 shows initial distribution of robots, Figure 4.21 and Figure 4.22 show the distribution after 1000 and 2000 steps respectively (each step represents around 200ms, there are totally 3200 steps, so the whole data last 640s, a little more than 10 minutes), Figure 4.23 shows final distribution. Figure 4.24 shows the trend of average distance change, which reaches equilibrium a few steps before 500.

The results show that robots spread out satisfactorily, from close initial distribution to looser final distribution, robots maintains connection as a team and reach setting desired distance eventually, and the final configuration is even and well organized. The only problem is the big fluctuation of the average distance, the reason, in our minds, comes from several aspects. Firstly, communication delay slows robots' response to distance change, and it takes time for robots to react to updated neighborhood. Secondly, dumping coefficients are partly responsible for the fluctuation. As a second order control method, our algorithm is very sensible to dumping coefficients selection. As a result, in order to solve this problem, in the future work, our work also aims on these two aspects, the first is to optimize communication setting and make the communication as smooth as possible, the second one is to adjust dumping coefficients manually through more experiments until satisfied results are obtained.

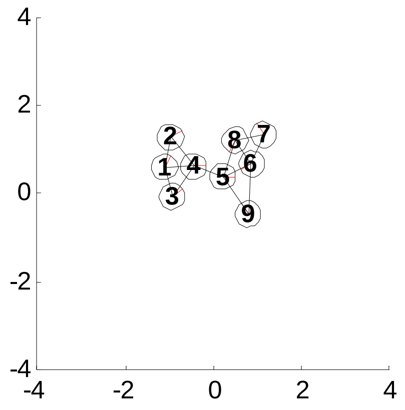


Figure 4.20: Initial distribution

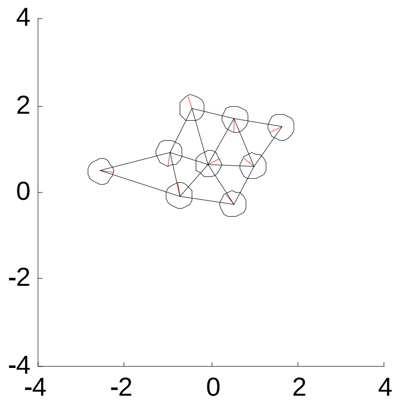


Figure 4.21: Distribution after 1000 steps

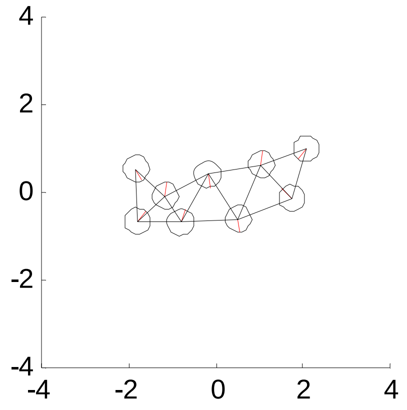


Figure 4.22: Distribution after 2000 steps

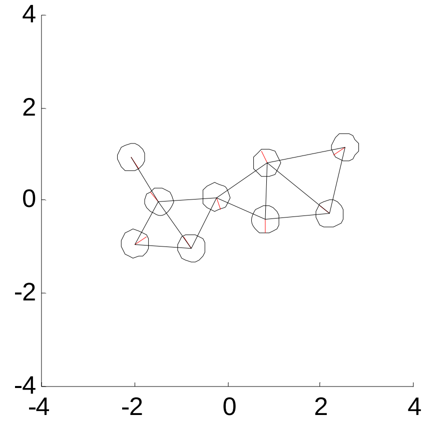


Figure 4.23: Final distribution, 3200 steps

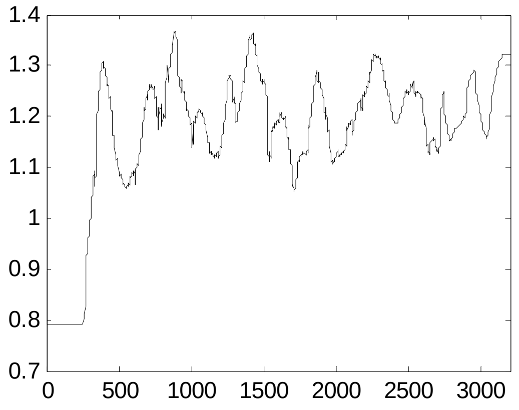


Figure 4.24: Convergence of average distances

4.4 Collision Avoidance Scheme II – Layer by Layer Spreading Out + Slowing down

In addition to the first collision avoidance scheme, which uses slow down to reduce and eliminate collisions, the second method combines a so-called layer by layer spread out scheme with the previous slow down. The name of this method implies how it works: robots on border will move first to outer space under potential forces, when space is left out, inner robots will start moving in turn. This method emphasizes in-order movements. Before enough space is left out by outer robots, inner robots keep stationary. This method effectively reduces chance of collision and on the other hand saves a lot of unnecessary movement in beginning stage of experiment.

One challenge in this method is lack of global information of all robots' positions. In other words, robots need to know its relative position in team with limited knowledge of its local neighbors. We thus propose a method to calculate every robot's relative position in the whole team using merely local neighbors' information. Simulation results show effectiveness of this method.

"Layer of depth" is used to mark robot's position with respect to team's border, the bigger it is, the deeper it is inside the team. The first step in our algorithm is to initialize all robots' "layer of depth" to '0'. Then robots are checked one by one to decide if it is on border or not. The method is to decide if all its neighbors are in the same semi-circle. As shown in Figure 4.25, for robot R_1, it has four neighbors, R_2, R_3, R_4, R_5. Fortunately, these four neighbors are all in the same semi-circle in R_1's neighborhood, so we set R_1's depth of layer to be 1. Otherwise, its layer of depth remains '0'. For robots that are not on border, its layer of depth depends on its neighbors, it communicates with its neighbors and find the one with the smallest layer of depth, its own layer of depth is this neighbor's layer of depth plus one. The algorithm keeps updating robots' layer of depths from time to time and eventually all robots have a reasonable layer of depth.

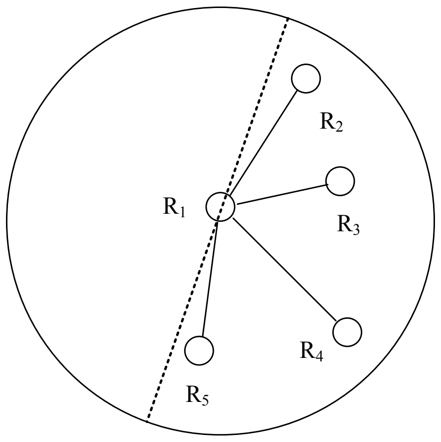


Figure 4.25: All four neighbors locate in the same semi-circle

The critical point of this method is to calculate layer of depth of robots on the border correctly. Sometimes, if we stick to the above principles strictly, we will find out the results are not as satisfied as we wanted. For example, in Figure 4.26, the robots on corners are recognized and marked '1' as their layer of depths. But for some other robots, for example, the robots on four sides of the team, their layer of depths is 2 while in our minds, it makes more sense if they have layer of depth of 1, because as we see, they are indeed located on the border. The reason they are marked in the second layer is that their neighbors are not strictly located in a semi-circle, instead, they are located in a sector slightly bigger than a semi-circle. Due to this phenomenon, we added a parameter called "maximum sector angle" in our method of calculating layer of depths. This parameter defines angle of the sector that all a robot's neighbors belong to. We can adjust this angle to be slightly bigger than 180 such that the robots mentioned above would have layer of depth of 1 instead of 2. The results of the new calculation are shown in figure 4.27.

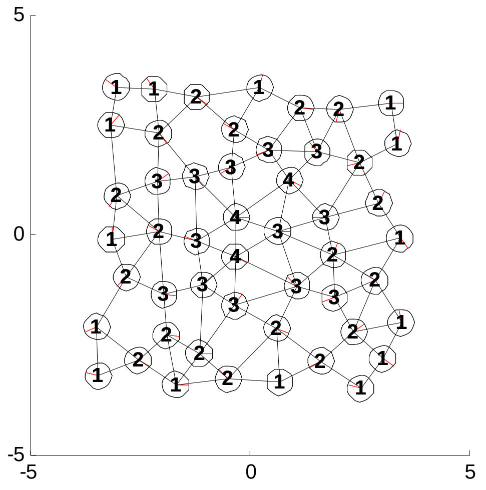


Figure 4.26: Layer of Depth when maximum sector angle equals to PI

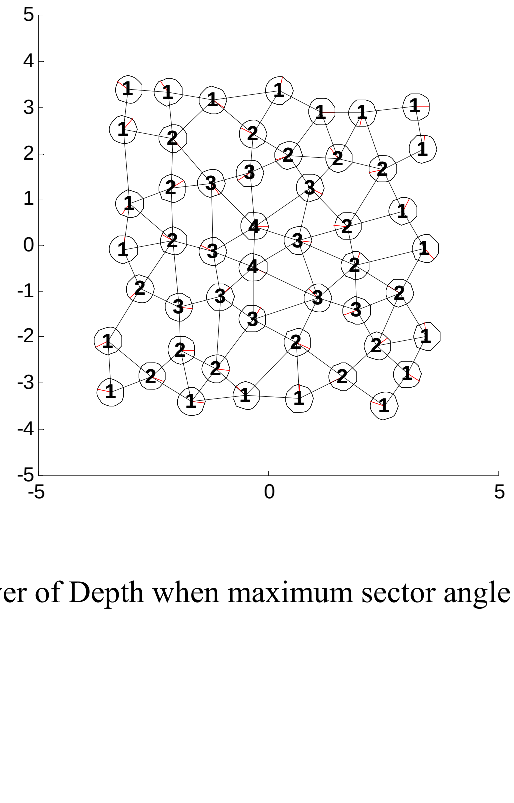
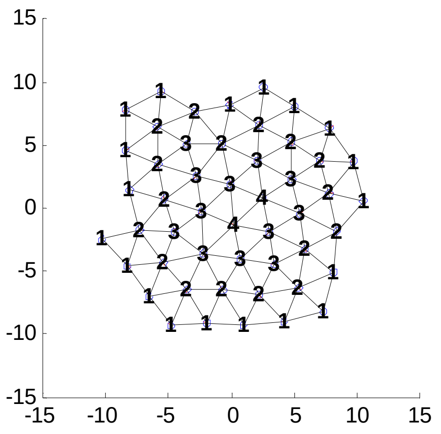


Figure 4.27: Layer of Depth when maximum sector angle equals to 1.2*PI

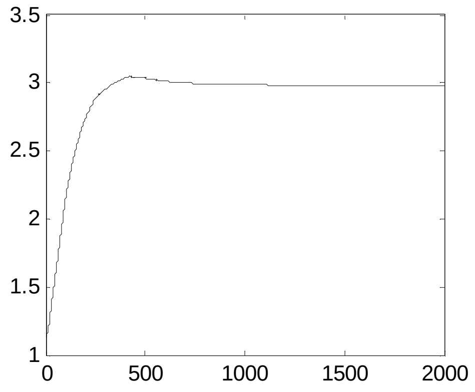
4.4.1 Results for Holonomic Robots

Figure 4.28 and Figure 4.29 show results for holonomic robots deployment using slowing down and layer by layer spreading out combined strategy. All parameters remain the same as described in section 4.3.1, except that for maximum sector angle, we set it to be 1.2*PI, which makes the layer depths calculation more reasonable and acceptable. The final outcomes of control laws 4.7 and 4.8 are shown in figure 4.28 and 4.29.

From the figures, we can see that robots are distributed evenly and inter-distances are well maintained. From the figure, layer depths of all robots are marked beyond each robot's icon. From our human being's mind, the layer depths are very close to our judgment and it proves that our distributed layer depth calculation method is working on all robots. Furthermore, the convergence of average distance also meets our expectations, and it reaches equilibrium quickly and stably. The steps takes to reach equilibrium is listed in table 4.6.

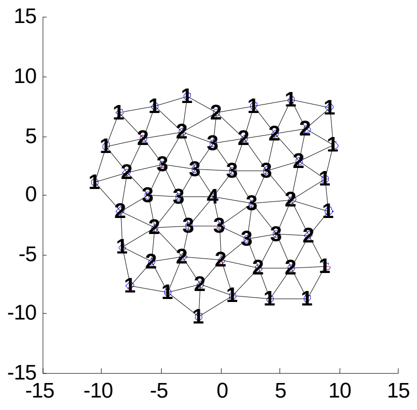


(c) Equilibrium team

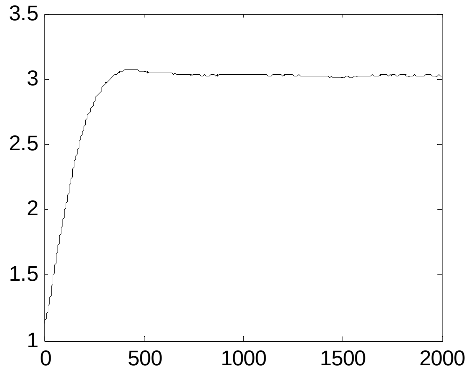


(b) Convergence of average inter-neighbor distance

Figure 4.28: Deployment simulation with control law (4.7) and layer by layer spreading out strategy



(d) Equilibrium team



(b) Convergence of average inter-neighbor distance

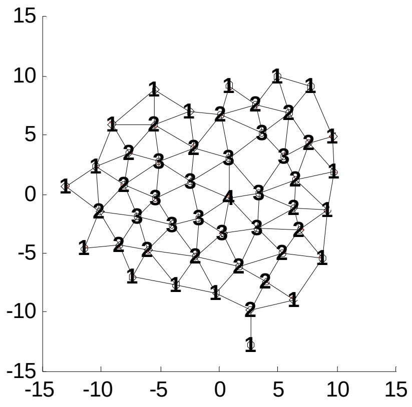
Figure 4.29: Deployment simulation with control law (4.8) and layer by layer spreading out strategy

Table 4.6: Rate of convergence

Control law	Collision avoidance strategy applied?	Settling time (steps)
(4.7)	No	171
(4.7)	Yes	231
(4.8)	No	148
(4.8)	Yes	273

4.4.2 Results for Nonholonomic Robots

The results for nonholonomic robots are slightly different from results of last section. Figure 4.30 and 4.31 shows results of the new collision avoidance schemes with control law 4.7 and 4.8. The final distributions are not so organized as we got in holonomic robots. The main reason of this is due to nonholonomic constraints. As robots in outer layers move outwards, they usually move overly further from the position that it is supposed to be, due to its inflexibility to move omni-directionally. After then, when they try to move backwards, it also takes time to adjust direction. All these adjustments result in not so even final configuration. Fortunately, robots eventually reach desired configuration and all robots are well connected. Steps required to reach equilibrium is also listed in table 4.7.


(e) Equilibrium team

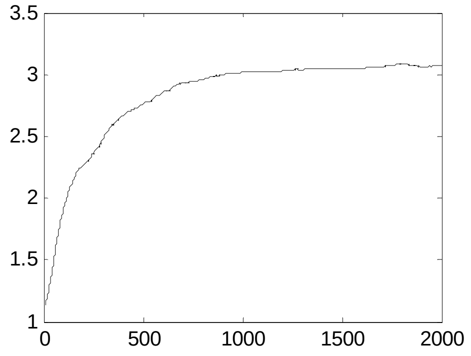
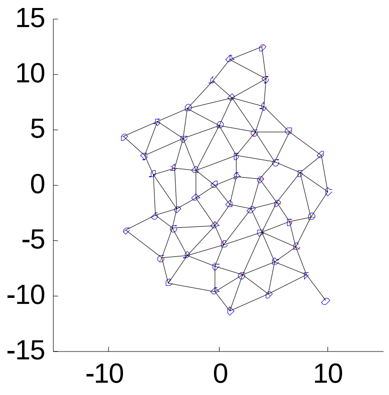
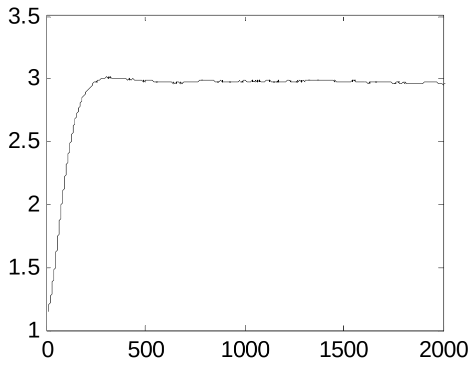

(b) Convergence of average inter-neighbor distance

Figure 4.30: Deployment simulation with control law (4.7) and layer by layer spreading out + slowing down



(f) Equilibrium team



(b) Convergence of average inter-neighbor distance

Figure 4.31: Deployment simulation with control law (4.8) and layer by layer spreading out + slowing down

Table 4.7: Rate of convergence

Control law	Collision avoidance strategy applied?	Settling time (steps)
(4.7)	No	655
(4.7)	Yes	668
(4.8)	No	562
(4.8)	Yes	172

4.4.3 Energy Consumption Saving with Layer by Layer Spreading Out

The most beneficial feature of using layer by layer spreading out strategy is that it greatly eliminates unnecessary movements and significantly reduces energy consumption, which is very important considering current battery power storage limit. With the new layer by layer spreading out strategy, robots have longer working hours and are able to provide better services on real applications.

The method we are applying here for energy consumption calculation is based on total distance the whole team has moved and the total angle the team has rotated altogether. In other works, the total energy consumption consists of two parts, linear moving energy consumption E_l and angular moving energy consumption E_a, total energy consumption is:

$$E = E_l + w_a * E_a \tag{4.20}$$

Where w_a is weight that derived from simulations and used to justify fairness of our equation. This will be discussed later in this section.

E_l and E_a are calculated in this way. At each step, we collect each small straight trajectory that each robot has moved and take their square sum as the total linear energy consumption. So if we have 50 robots and the simulations run 2000 steps, we have:

$$E_l = \sum_{k=1}^{1999} \sum_{i=1}^{50} [(x_{k+1,i} - x_{k,i})^2 + (y_{k+1,i} - y_{k,i})^2] \tag{4.21}$$

Where $(x_{k,i}, y_{k,i})$ and $(x_{k+1,i}, y_{k+1,i})$ are current and next step's coordinate of robot i, the unit is meter.

Similarly,

$$E_a = \sum_{k=1}^{1999} \sum_{i=1}^{50} (\theta_{k+1,i} - \theta_{k,i})^2 \tag{4.22}$$

Where ($\theta_{k,i}$, $\theta_{k,i}$) and ($\theta_{k+1,i}$, $\theta_{k+1,i}$) are robot i's direction at current and next step. The unit is radian.

In order to get a more convincing result, each simulation is run 12 times with 12 different set of robots with totally different initial positions and directions. However, before that, we need to get the value of w_a. For the calculation of weight w_a, we run a trial simulation for 2000 steps and obtain values of E_l and E_a respectively, then w_a is equal to E_l divided by E_a. For example, for slowing down strategy only, we got E_l=15.1323 and E_a=13.8466, so w_a=15.1323/13.8466=1.0929. This makes sense because it makes linear and angular movements equally important in total energy consumption calculation.

After we got w_a, 12 different sets of robots with different initial positions and directions is fed in the algorithm. The generation of these initial robots is discussed in Section 4.2.2. As a result, for algorithm using only slowing down strategy, the eventual average energy consumption is 27.1668, listed in Table 4.8.

For combined collision avoidance strategy (layer by layer spreading out strategy is included), E_l= 9.2785, E_a= 6.9163 and w_a=9.2785/6.9163=1.3415. Total average energy consumption is 19.2033.

Table 4.8: Energy consumption comparison

Control law	Slowing down strategy only	Combined strategy, adding layer by layer spreading out
(4.7)	27.1668	19.2033

From the simulation, we can easily find out that layer by layer spreading out strategy has successfully reduced energy consumption by 29.3%, which is almost 30%. The result is very impressive because that means with reaching the same final configuration and finishing the same environment coverage task, robots with layer by layer spreading out strategy could work 30% longer. If robots with regular slowing down strategy work normally 10 hours, robots with layer by layer spreading out strategy could work up to 13 hours.

4.5 Conclusion

Targeting robust deployment of multiple mobile robots under realistic constraints, this chapter has proposed a highly effective distributed control scheme for reliably accomplishing desired sensor coverage among a team of nonholonomic mobile robots. Besides, the collisions avoidance is also addressed and two schemes are proposed. The effectiveness of the proposed scheme has been verified through both computer simulations and experiments on a team of physical robots.

Global convergence towards the desired coverage is an important property of multi-robot deployment. We have reported some simulation and experimental results about it in this chapter, but have not comprehensively and theoretically study it yet. Our next step will focus on studying and analyzing the convergence property of the multi-robot deployment towards the desired coverage and the impact of multiple factors arising from physical systems, such as robot state estimation error, communication delay and operational asynchrony among the robots, on system convergence.

Chapter 5 Robotization of MMT

Energy crisis has been a big challenge in our society nowadays. Besides exploring alternative clear energy like solar and wind power, it is also important to conduct in-depth analysis of energy consumption and improve its efficiency. Data centers have been energy monsters along with development of information technology. Actually, in 2005, United States's data centers, including power consumption by servers, network, cooling system and other relevant facilities, consumed 1.5 percent of total electricity use [59]. This chapter will discuss an energy usage analysis system deployed in data centers.

This so-called MMT (Mobile Measurement Technology) for data centers is constructed by IBM. It provides highly effective data support for the analysis of data center thermal profiles and plays an active role in improving data center cooling and energy efficiencies. With this technique, a human operator controls the navigation of the tele-operative mobile measurement platform and triggers the data acquisition operation through a wirelessly connected console. Comparing with traditional manual operation, it largely reduces the intensity of human labor and improves the operational efficiency. The effectiveness of the proposed technique has been verified through a demonstrative data scanning in a real data center environment.

5.1 Motivation and Old MMT

5.1.1 Requirements and Objectives of MMT

Data centers provide critically important computing capabilities, including data processing, data storage and communication networking, to the functioning of business, communications, academic and governmental organizations [59]. To maintain the operational reliability and availability of data centers, air cooling plays an important role. In a standard data center setting, IT equipment is mounted in racks that are positioned side by side in long rows, and rows of racks are separated by alternating hot and cold aisles (Figure 5.1). The racks are placed on a raised floor, which allows the conditioned air to be delivered from the bottom to remove the hot air from the top. Electronic equipment in such a confined space generates a significant amount of heat, and the equipment's reliability is reduced if it is not adequately cooled. Inappropriate humidity levels can also cause the failure of electronic components in data centers [60]. To maximize the functionality of a data center, it is necessary to keep the electronic equipment to operate within the temperature and humidity ranges specified by the manufacturers. It requires sufficient air conditioning across the data center space.

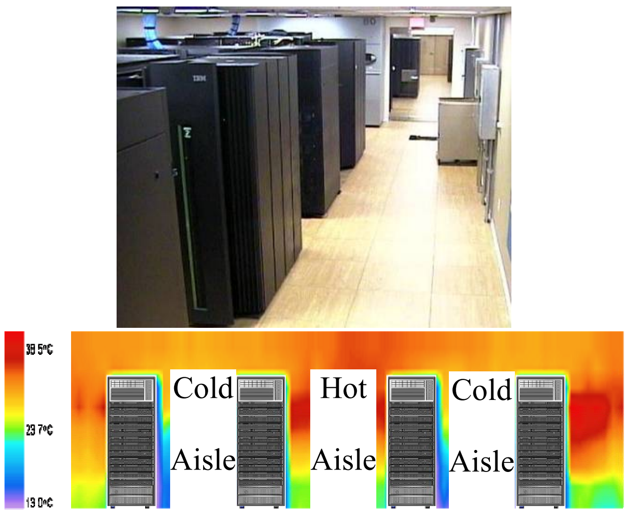


Figure 5.1: Typical equipment placement in data centers

Meanwhile, techniques to improve energy efficiency have become important for the data center industry [61]. A large amount of energy is consumed by data centers to run their computing, storage and networking systems, and peripheral devices, and to protect these systems. It was estimated [59] that U.S. data centers consumed about 61 billion kWh of electricity in 2006, about 1.5% of total U.S. electricity consumption. To support the growing demand for processing power, the total number and size of data centers continue increasing, and individual data centers increasingly use more compact and energy-intensive servers. This growth in the data center industry causes a dramatic increase in electricity demand. It was predicted [59] that the total energy consumption in U.S. data centers would go beyond 100 billion kWh by 2011. The increasing energy consumption in data centers causes increasing energy costs for business and government, emission of greenhouse gases, load on existing power grid, etc. All these factors have driven recent interest in improving energy efficiency of data centers [61]. As a critical but energy-intensive component, air conditioning systems have been identified as one of the main energy consumers in data centers. It was estimated that a fully populated rack of servers could generate 20-25 kW of heat during operation [62], which requires at least 20-25 kW of output cooling power from air-conditioning systems. Improving the air flow will substantially improve the overall energy efficiency of data centers.

In summary, an optimal air-conditioning scheme for a data center should achieve a safe operating conditions for IT equipment with the highest possible energy efficiency. In order to accomplish this double-folded goal, the data center's thermal profile needs to be measured and analyzed. Quantitative thermal profile of the data center will highlight air-conditioning inefficiencies and insufficiencies, such as cold and hot spots. In-depth analysis of these trouble spots and corresponding metrics of air-conditioning and energy efficiencies can lead to optimal air-conditioning schemes which optimize the energy and air-conditioning levels across the data center space, and eliminate the spots of insufficiency and over-provision. The thermal profile will also allow an operator to monitor the operational status of the data center and to evaluate the improvement in air-conditioning efficiency after a new scheme has been implemented.

A major challenge in quantifying and visualizing the thermal profile for a data center is reliable data collection. Especially, to create a complete 3D thermal profile for a data center requires collecting temperatures across the whole 3D data center environment, including not only the points on solid surfaces where static sensors could be attached but also the points in the air where no sensor can be placed. Targeting to provide an effective 3D data collection solution in the data center environment, IBM recently developed a Measurement and Management Technology (MMT) which uses a multi-level cart equipped with networked thermal sensors to acquire temperatures at multiple points in 3D [63-65] (Figure 5.2). The sensing platform has a footprint of the size of a standard tile (2 feet×2 feet) used to cover the raised floor of data centers. It samples the temperatures at multiple points above each tile. The collected temperature data are then transformed into a 3D thermal map of the data center, which provides the vital information needed to pinpoint trouble spots that indicate cooling inefficiencies, to facilitate better air-conditioning schemes, and to manage the energy consumption of the data center.

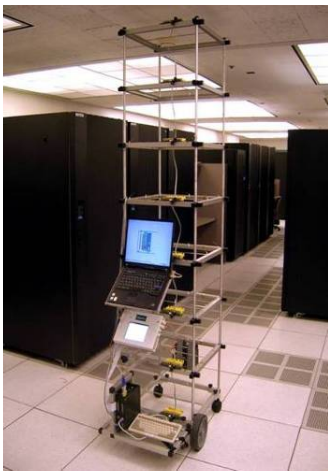


Figure 5.2: IBM MMT sensing cart

Although the thermal profile of a data center may change over time due to temporal changes in IT power level, cooling condition, number of servers and racks, etc., the MMT technique enables a high-density collection of temperature data for basic modeling of the data center thermal profile. If more precise, dynamic modeling is needed, a static sensor network can be installed to monitor the temporal variations of the thermal conditions in the data center. In this case, the basic thermal profile generated from the MMT data can be used to identify critical spots for sensor placement. Moreover, due to the limitation that sensors can only be attached to solid surfaces but not in the air, the basic thermal profile generated from the MMT data provides an important reference thermal profile to facilitate dynamic modeling based on interpolating the sparse temperature data captured by the fixed sensors.

5.1.2 Implementation

A typical MMT consists of a cart with 8 levels of sensors (Figure 5.1 and Figure 5.4) (Each level of the cart has 9 sensors), an Interface Box(Figure 5.2), a handler(Figure 5.3) and a laptop. The Interface Box is wired with the handler, the sensors and the encoders. The other side of the Interface Box is a laptop, and they are connected via a RS232-USB cable. Interface Box plays a bridge between sensors and laptop. On the one hand, it processes raw data from sensors and converts it into numeric numbers, and laptop accepts data and saves it to local hard drive. On the other hand, it also accepts signal from handler or laptop and transmit it to sensors.

The data collection process starts from the handler. Once its button is pressed, it generates an impulse, this impulse reaches interface box and interface box triggers sensors to collect data and collects current signal from encoders accordingly. After that, the interface will convert raw data into numeric numbers and send them back to the laptop, which will update the cart's current orientation and position, and save data along with positions in local hard drive.

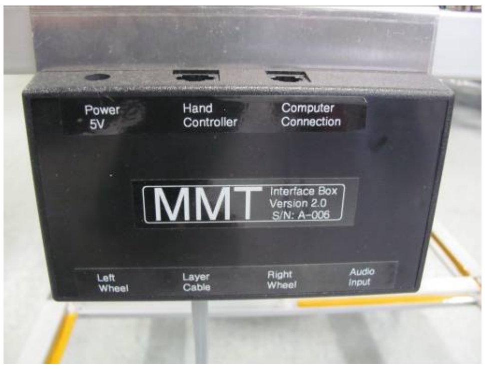


Figure 5.3: Interface Box

Figure 5.4: Handler

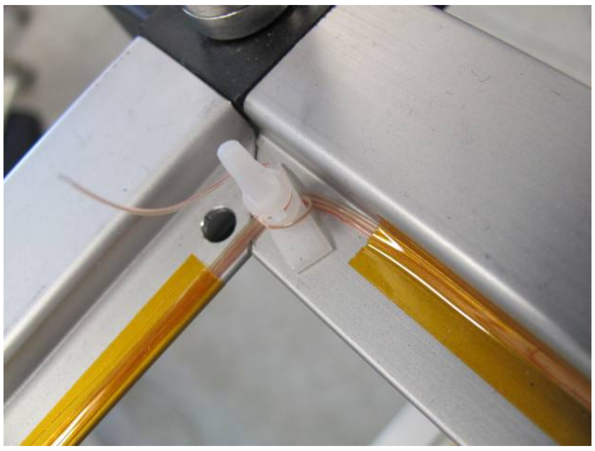


Figure 5.5: Temperature sensor

In order to process data from Interface Box, a program named Tmapr (Figure 5.6) is developed. It is able to load given layout of the target data center, update cart's orientation and position according to received data, and save data to local disk drive. In Figure 5.6, red icon with shape of "V" represents the cart, the smaller upper level window displays current sensor data. The red icon can be moved and rotated manually by using keyboard's arrow keys to reflect the cart's real location and orientation.

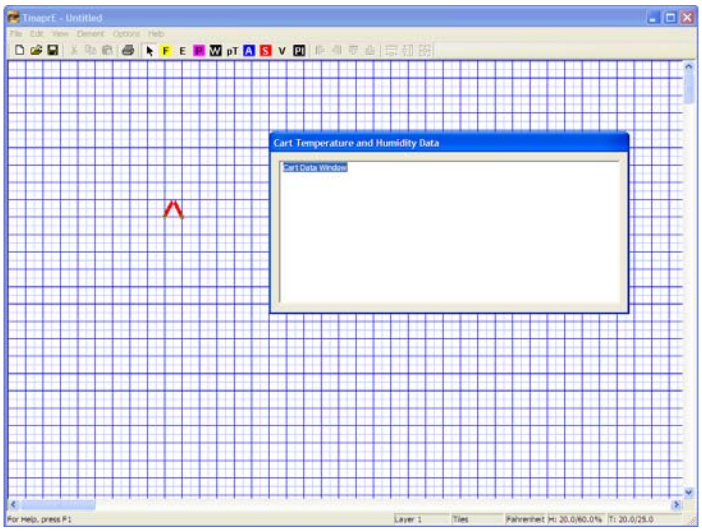


Figure 5.6: Tmapr interface

5.2 Robotic MMT

As we mentioned in the previous section, the current version of MMT platform is operated manually. It requires an operator to push the cart to cover every tile of the data center floor. The cart is stopped at each tile, and data acquisition is manually triggered to register the temperature data obtained from onboard thermal sensors and the corresponding cart location (in the unit of tiles) inferred from the reading of wheel encoders. Thus, currently the data center scanning process is labor-intensive and time-consuming, typically taking one hour for a data center of 2000 square feet. In order to improve the operational efficiency of MMT, we propose to develop a robotic mobile sensor platform for data center navigation and measurement to automate the data acquisition process, which is named as Robotic Measurement and Management Technology (RMMT).

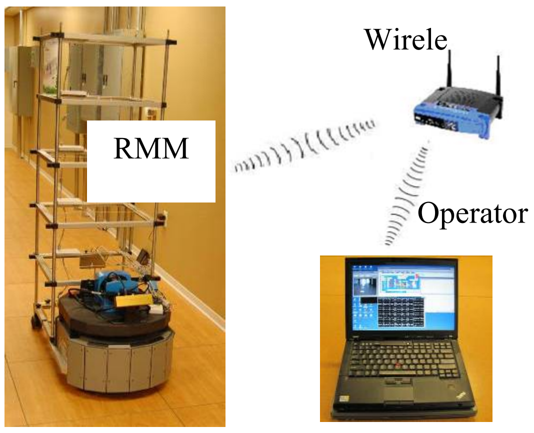


Figure 5.7: New MMT under work

The robotization of MMT brings in the following immediate benefits:

1) The robotic system enables automatic drive and triggering of the sensor platform, which greatly saves human labor and makes it more available to provide quick response and assessment to significant variations in a data center.

2) The robotic system enables continuous, robust data collection over a long period up to its power limitation, which substantially enhances the efficiency and reliability of operation.

Perspectively, it will also extend the MMT functionality in the following aspects:

1) The robotic system will provide the capability of quick and high-resolution data acquisition for temperature, humidity, air flow and other environmental data, with appropriate sensors onboard.

2) The robotic system will provide the capability of onboard generating the thermal profile, or wirelessly uploading data to MMT servers for real-time processing and visualization.

3) The robotic system will be able to conduct map-based autonomous navigation, or autonomous, simultaneous mapping and navigation when the data center layout is not known a priori, which will eventually release humans from operation, and maximize the efficiency of data collection.

4) The robotic system will be able to work collaboratively with any sensor network in the data center, and supplement the measurements for more accurate dynamic modeling.

This section presents a preliminary, tele-operative version of the RMMT platform. This system uses a mobile robot base carrying cameras and localization sensors for navigation and localization. The robot pulls an MMT cart which carries thermal sensors for temperature measurement. The robot base is tele-controlled by an operator using a remote computer based on the wirelessly transferred images captured by onboard and environment-fixed cameras. The data of temperatures and locations are automatically registered tile by tile and stored in the onboard computer. The collected data are then input to IBM's MMT data processing software to generate the thermal map of the data center. A demonstrative test of the RMMT system has been conducted in a real data center environment. The robotized MMT platform provides a more efficient and user-friend technique for the targeted application.

5.2.1 System Structure

The current version of RMMT system (Figure 5.6) is a tele-operation system, consisting of two physically separate but wirelessly connected subsystems:

1) Robotic navigation and sensing platform, which conducts the tele-controlled navigation through the data center and collect temperature data on the way;

2) Operator's console, which provides a user interface for the human operator to tele-control the navigation and data acquisition of the RMMT platform.

5.2.2 RMMT Platform

The robotic navigation and sensing platform is the tele-operator in the RMMT system.

Hardware-wise, the RMMT platform consists of a mobile robot base and an MMT sensor cart (Figure 5.6).

The mobile robot enables the RMMT platform to conduct the controlled navigation through the data center. A PatrolBot manufactured by Mobile Robots Inc. is adopted, due to its 12-kg carry-on payload allowance and 9.1-kg pulling payload allowance which are sufficient to drive the MMT sensor cart [66], and its decent collection of onboard sensors which will facilitate future functional extension of the system. An embedded onboard computer communicates with the operator's console through wireless Ethernet (compatible with IEEE 802.11a/b/g/n protocols), and controls the robot motion based on the operator's commands. The readings of the wheel encoders are used to localize the robot and cart. The onboard cameras (Bumblebee dual-camera stereovision unit) provide vision feedback for navigation guidance and obstacle avoidance.

The MMT sensor cart carries thermal sensors (standard K-type thermocouples are currently used) to measure temperatures. The MMT cart has a layered, stackable design, with an equally-gapped 3×3 sensor arrangement in each layer and 1-foot gap between neighboring layers, resulting in a 1-cubic-foot data resolution which is highly sufficient for generating accurate 3D thermal profiles for data centers. Due to its stackability, the height of the MMT cart can be adjusted to fit with a wide range of ceiling heights. With this design, the MMT sensor cart can simultaneously measure temperatures at different heights above each tile, which makes data acquisition highly efficient. The cart is rigidly attached to the mobile robot, and supported by the chassis of the mobile robot and 2 rear swivel casters attached to the cart, such that the cart is smoothly driven by the robot and can be easily localized according to the position and orientation of the robot. The onboard thermal sensors are networked through an integrated programmable microprocessor interface. Their readings are registered using a data logger, further transferred to the onboard computer through an RS-232 serial interface, and stored in the computer.

The MMT data acquisition and robot navigation are synchronized. In a data acquisition operation, the robot pulls the MMT cart to move a distance of one tile for each step, such that the cart can seamlessly cover every tile eventually; then the robot makes one stop, and triggers the data logger on the cart to register the current readings from the thermal sensors; the onboard computer receives the temperature data from the data logger and combines them with the current position of the cart to provide a temperature-location data set for the current tile; the data sets collected from all the tiles will be used to generate the 3D thermal map of the data center.

A functional diagram of the onboard RMMT software system can be found in Figure 5.8. The onboard RMMT controller monitors and coordinates the functioning of three modules – navigation, sensing and communication.

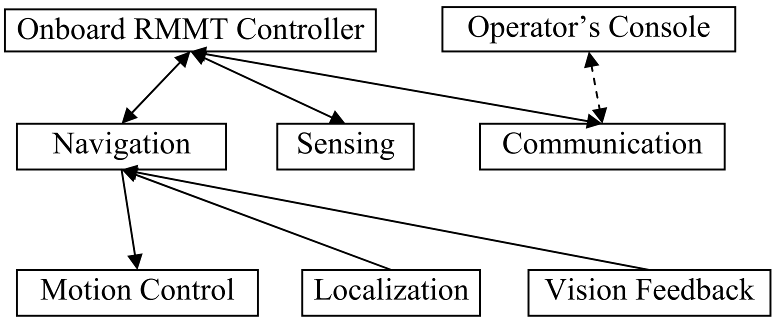


Figure 5.8: Functional diagram of the onboard RMMT software

The navigation module consists of the following units:

1) Motion control unit: It translates the operator's motion commands into the corresponding instructions for translational and rotational movements of the robot, and instructs the lower-level robot controller to implement the desired motion.

2) Localization unit: It takes the readings of the wheel encoders of the robot, translates them into the position and orientation of the robot, and then localizes the sensor cart. Since the sensor cart is rigidly attached to the robot, the orientation of the cart is same as that of the robot, and the position of the cart is obtained by displacing that of the robot along the robot's orientation.

3) Vision feedback unit: It captures the images of the floor and objects nearby the robot using the onboard video cameras, and sends them to the operator's console wirelessly in

real time. Based on the vision feedback, the operator can command the robot to move in the desirable direction, avoid collisions, and compensate the localization errors.

The sensing module mainly consists of a data acquisition unit which, upon being triggered, collects the temperature data from the networked thermal sensors on the sensor cart.

The communication module receives commands from the operator's console and sends the visualized results of onboard sensory feedback to the operator's console through the wireless Ethernet link between the onboard computer and the operator's console.

At any moment, the onboard RMMT controller works in one of the following two modes, following the operator's command:

1) Data acquisition mode: In this mode, the RMMT controller instructs the navigation module to move the sensor cart either one tile forward or one tile backward, and then to let the robot and cart make a stop. Next, it registers the current position and orientation of the cart. Because the temperature data are collected tile by tile, the RMMT controller registers the cart position and orientation by snapping the position into the tile and orientation into one of the four principal directions, i.e. forward, backward, left and right with respect to the tile. Then, it triggers the data acquisition unit to capture the temperature data above the tile. In this way, the temperature data are registered with the tile location, which provides a complete data set for the space above the specific tile.

2) Navigation adjustment mode: In this mode, the RMMT controller instructs the motion control unit to turn +/-90 such that the cart changes its navigational direction, or to move the sensor cart forward/backward and left/right in small steps to align the cart to the desired position and orientation in the current tile in order to compensate the localization inaccuracy due

to the accumulated error from the encoder-based odometry. In this mode, the RMMT controller does not trigger the data acquisition operation.

By conducting data acquisition tile by tile, the RMMT controller will eventually collect a complete set of temperature-location data from which the thermal map of the data center can be generated.

5.2.3 Operator's Console

The operator's console is the user interface between the human operator and the tele-operator –the RMMT platform (Figure 5.2). It mainly consists of a console program running on a remotely-located computer (e.g. a laptop PC), and includes the following modules:

1) Commanding module: It enables the operator to input navigation and data acquisition commands to remotely control the RMMT platform. The control commands are input using the keyboard of the console computer. The current set of commands include two combined navigation / data acquisition commands, i.e. "move sensor cart forward to next tile, and then acquire data" (the "f" key) and "move sensor cart backward to next tile, and then acquire data" (the "b" key), and six simple navigation commands, i.e. "turn robot left 90° in the current tile" (the "l" key), "turn robot right 90° in the current tile (the "r" key), "turn robot slightly left" (the "left arrow" key), "turn robot slightly right" (the "right arrow" key), "move robot slightly forward" (the "upward arrow" key), and "move robot slightly backward" (the "downward arrow" key). Being self-explanative, these commands provide the basic set of functions for the operator to control the navigation and data acquisition of the RMMT platform. The commanding module issues the operator's commands to the onboard RMMT controller program. The simple navigation commands trigger the RMMT controller to work in the navigation adjustment mode,

taking into effect through the onboard motion control unit; the combined navigation / data acquisition commands trigger the RMMT controller to work in the data acquisition mode, taking into effect through the onboard motion control unit and sensing module.

2) Feedback display module: It provides the visualized feedback of the cart position/orientation and the environment such that the operator can make appropriate decision on steering the RMMT platform and acquiring data. It consists of two basic feedback windows: the localization window which visually outputs the results of the onboard localization unit, i.e. current position and orientation of the sensor cart in the data center layout (Figure 5.9), and the vision window which displays the real-time images captured by the robot's onboard cameras (Figure 5.10).

3) Communication module: It transmits commands from the console's commanding module to the onboard RMMT controller and receives the visualized results of onboard sensory feedback to the console's feedback display module through the wireless Ethernet link between the operator's console and the onboard computer.

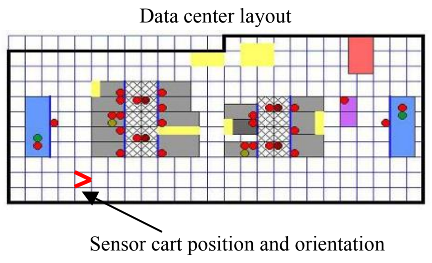


Figure 5.9: Sensor cart localized in the data center

Figure 5.10: Feedback from onboard camera

In summary, in the current version of RMMT, the operator's console functions as the hyper-terminal of the onboard RMMT controller. The commands input from the console are directly delivered to the onboard controller to trigger the corresponding navigation and data acquisition behaviors, and the localization and vision feedbacks obtained onboard are also directly delivered to and displayed on the console.

Under the situation of one human operator controlling one tele-operator, the hyper-terminal operation mode provides a convenient solution to the targeted tele-operation application. To maintain robust data communications, particularly for real-time image delivery, between the operator's console and the RMMT platform, a simple ad hoc wireless link is established based on the IP addresses of the console and the onboard computers, using a wireless Ethernet router (compatible with IEEE 802.11a/b/g/n protocols, and having a reliable communication range of >100 feet). As a result, the data center scanning operation does not require any access to local wireless network, which makes it easy for the RMMT system to be set up for different data centers, particularly for those which impose highly restrictive, protected network access due to the concerns of data and network security.

5.3 Hardware Components of RMMT

Instead of control MMT's movement directly, a third party commercially available mobile robot platform-PatrolBot is used and mechanically integrated with MMT to provide moving force. PatrolBot is developed by MobileRobots Inc., and it have maximum load of 12 kilogram and can reach pushing force of 9.1 kilograms [54] and therefore is completely capable of pulling the MMT.

Figure 5.11 is the mechanical joint between PatrolBot and MMT.

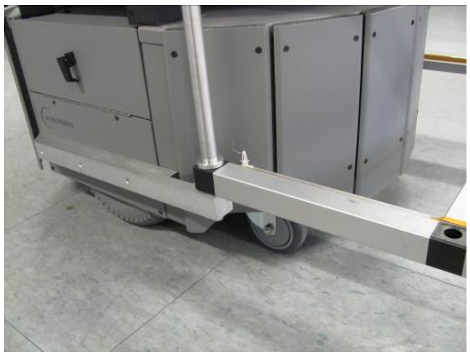


Figure 5.11: Mechanical integration between MMT and robot

Next we use a separate laptop to connect with PatrolBot's onboard computer, send command to the robot and get real time camera feedback. The onboard computer is also responsible for connecting with MMT and recording sensor data. In order to reach remote operation, a wireless router is required. In our work, we use Linksys wireless router and it has communication range of 100-300 feet. If Patrol's onboard computer is connected to Internet directly, theoretically we can operate the robot from home.

So basically, the new system consists of four parts, MMT, PatrolBot with an onboard computer, a wireless router and a laptop with wireless adapter.

Besides, the camera feedback we are using with the robot is a dual-camera Bumblebee stereo vision system. Human being is able to observe obstacles and layout of the environment, and makes robot move in a well-planned routine. In other words, the onboard camera provides necessary local and global views of the data center for obstacle avoidance and path following.

Figure 5.12: User Interface

Besides, the onboard computer of PatrolBot is connected with MMT's Interface Box via a data board and a RS232-USB cable. The cable transmits data including sensor data and mimicking encoder pulses to onboard computer, and the computer sends mimicking encoder pulses and data acquisition triggering signal to the Interface Box through a NI data board. It is strange that the computer sends mimicking encoder pulses to Interface Box and it sends the same signal back to onboard computer, but it is the way how the new MMT works, the reason is, as we mentioned earlier, to make the system compatible with the old MMT system. In other words, the new improvement could be finished on any other MMT without doing any change to it.

5.3.1 Data Acquisition Triggering

Figure 5.13: NI USB 6501

Instead of pressing handler button every time to collect sensor data, in the new MMT system, we defined a key to finish this work. A data board is used to generate exactly the same signal as the old handler does, which is a duration of high voltage lasting 2 seconds.

NI USB 6501 data board fits the requirement well and is cost effective. It has 24 digital I/O lines and overvoltage protection. One digital output channel is occupied for this triggering purpose and another channel is reserved for mimicking encoder pulses, which we will discuss later.

5.3.2 Encoder Pulses Mimicking

As we mentioned earlier, Tmapr is responsible for collecting and displaying data. When robot is moving, it collects encoder pulses from wheels and calculates robot's position thereafter. However, this program was developed to recognize direct encoder pulses other than direct position information. Besides, in our future work, we don't deny the possibility of keeping using of encoder readings for better localization accuracy and moreover we hope our work can be applied to other MMTs quickly without too many modifications. Therefore, in the new system, in order to be compatible with Tmapr, mimicking encoder pulses are generated according to position change, which we obtained from PatrolBox's in-built position calculation. In other words, given initial position and orientation of the robot, robot is able to calculate its own position and orientation from time to time, our program reads out the location and direction change and converts it to a sequence of encoder pulses. The pulses are input to Interface Box and it sends back to computer via RS2432-USB cable, Tmapr is able to update MMT's position and orientation based on it. As a result, at every stop, Tmapr will firstly evaluate encoder pulses and calculate its current position and orientation with knowledge of its location information at last stop, and then record corresponding thermal data, match the data with locations and save them to local hard disk.

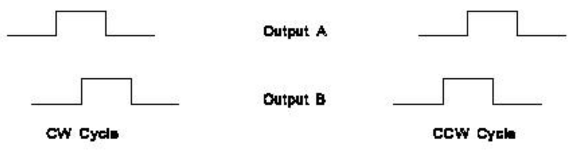


Figure 5.14: Encoder pulses

Figure 5.14 and Table 5.1 show mechanism behind calculation from encoder pulses to real physical movements. The MMT cart optical encoders output a 2-bit quadrature signal to indicate the direction and rotation amount of the encoder shaft. The A and B signals as show in Figure 5.14 indicate the direction of the encoder shaft, the 2 signals are 90 degrees out of phase of each other and the order of two sequences determines wheel's rotation direction, phase error with the two signals should be less than 45 degrees. That means if signal A is 130 degrees ahead of signal B, the two signals are still considered effective and the wheel is rotating clockwise.

Based on Figure 5.14, there are 6 possible combinations of signal durations, 3 forward and 3 reverse directions, from which MMT's physical movements are calculated. They are follows:

Table 5.1: Movement meaning of encoder pulses

	Forward		Reverse	
	Left Wheel	Right Wheel	Left Wheel	Right Wheel
Move 1 tile	582 CW cycles	582 CCW cycles	582 CCW cycles	582 CW cycles
Left Turn	0 cycles	810 CCW cycles	0 cycles	810 CW cycles
Right Turn	810 CW cycles	0 cycles	810 CCW cycles	0 cycles

5.4 Results

The developed RMMT system has been used to scan IBM's Southbury Green Innovation Data Center to assess the feasibility of characterizing large data centers. In our knowledge, this is the world first reported robotic data acquisition operation conducted in a real data center environment.

IBM Southbury Data Center has a 2000 square-foot space, hosts more than 300 servers, has more than 100 terabytes of storage, and supports up to 200,000 users' online access. It is equipped with uninterruptible power supply (UPS), 200 kilowatt power distribution unit (PDU), and energy-efficient computer room air conditioning (CRAC) units.

To scan the data center, the human operator was situated in an adjacent room and monitored the tele-operative RMMT platform by watching the images of the environment displayed on the operator's console, and the RMMT platform was moved by remotely commanding it through the ad hoc wireless link. Besides the images captured by the onboard cameras (Section III), the images from three ceiling-mounted surveillance cameras were also displayed on the console (which were delivered to the console computer through the Internet) to further enhance the reliability of navigation and collision avoidance (Figure 5.15).

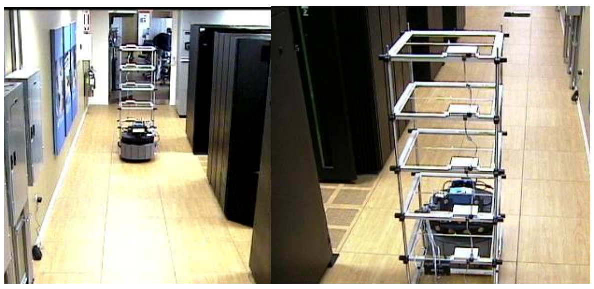


Figure 5.15: Images from environment-attached cameras

For navigation guidance, the data center layout served as a map (Figure 5.8), and was loaded on to the onboard RMMT controller. The data center was scanned through the following process (Figure 5.15).

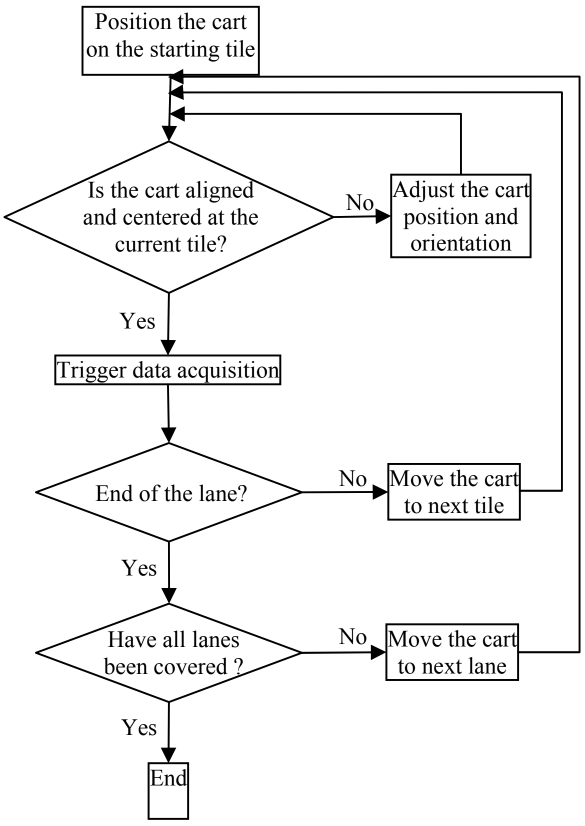

Position the cart on the starting tile
Is the cart aligned and centered at the current tile?
No
Adjust the cart position and orientation
Yes
Trigger data acquisition
End of the lane?
No
Move the cart to next tile
Yes
Have all lanes been covered ?
No
Move the cart to next lane
Yes
End

Figure 5.16: Flowchart of the data center scanning process

During this process, the global position of the sensor cart in the data center was determined by counting the numbers of horizontal and vertical tiles with respect to the starting tile. At each tile, the RMMT platform stayed for 3 seconds in order for the readings of the thermal sensors to stabilize before triggering the data logger. The temperature and location data were recorded onboard.

After scanning, the recorded data were processed using IBM's MMT software. The resulting thermal profile of the data center is shown in Figure 5.17. One can see that in general the left side of the data center layout has higher temperatures than the right side, which shows that potentially the cooling distribution can be improved to achieve a more uniform temperature profile and thus higher energy efficiency.

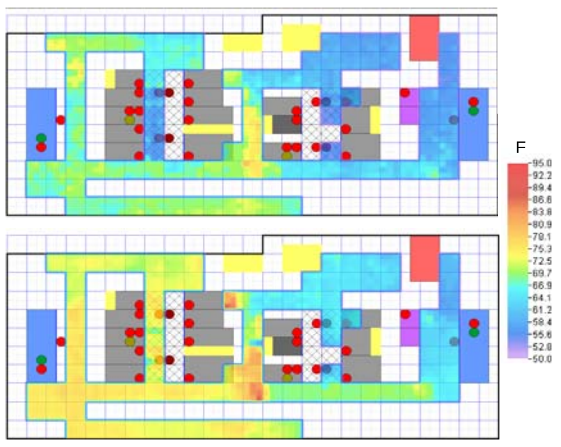


Figure 5.17: Maps of temperatures at 0.5 (upper) and 4.5 (lower) feet above the ground

5.5 Conclusion and Future Work

So far we have introduced our tele-operative RMMT system for measuring temperatures in data center environments, which provides an effective data acquisition tool for the analysis of data center thermal profile and the improvement of cooling and energy efficiencies. The demonstrative test of the current RMMT system in a real data center environment has proven the effectiveness of the proposed technique. In particular, the robotization of the MMT technique can reduce human labor and improve the data acquisition efficiency.

Robotizing MMT is a natural development step to enhance the capabilities of the existing MMT technique. Our ultimate goal is to take the full advantage of robotization to achieve continuous, fast and autonomous data acquisition process in any data center. To approach this goal, we have identified the following technical directions to further our exploration:

1) Improvement of the tele-operative RMMT technique: We will improve the user interface by developing joystick-based smooth navigation control and data acquisition triggering,

and developing more comprehensive onboard vision feedback system for more reliable and agile navigation planning and obstacle avoidance. We will offer the tele-operator more autonomy by enabling the RMMT platform to self-align to the tile and compensate motion and localization errors based on onboard vision and ranging feedback. Moreover, we will further expand the sensing capability of the RMMT platform by incorporating appropriate sensors to enable the measurement of environmental parameters other than temperature.

2) Development of the fully autonomous version of RMMT: We will integrate autonomous navigation and environment mapping capabilities into the RMMT platform to minimize human involvement, maximize operational efficiency, and make it adaptable to different and evolving data center layouts.

Chapter 6 Future Work

Besides the work we have done, following is the list of work we are going to finish in the future:

1) Multi-robot collaborative localization based on camera and encoders

Localization is always a problem for both single and multiple robots. Though our single camera based localization algorithm is accurate, it is hard to guarantee that at any moment there are at least 3 landmarks in view of the camera. As a result, we need to combine several localization methods together. For example, in our case, data from encoders and camera could be combined. To sum up, a scheme that takes advantages of camera and encoders is desirable.

The benefit of using encoders is that it is fast and cost effective. However, they have a big shortcoming that they have accumulative error. Compared with encoders, camera localization is a global localization method and is more accurate. However, compared with encoders, camera is more expensive and image processing always takes a lot of computing power and time. By combining the two sensors together, with help of camera, it is likely this accumulative error could be corrected from time to time. Moreover, the camera is not required to do localization in real time. It is just responsible for correcting accumulative error for the encoders on a regular basis.

2) Continued work on collision avoidance

So far our work on collision avoidance indicates how collision is reduced with bigger size of initial distribution area. Besides, adjustment of parameters, for example, damping coefficients and the resistant force we defined to slow down robots also play important roles in collision avoidance.

However, a complete collision avoidance scheme is still desirable. This scheme should work effectively despite of the initial distribution of robots. “Effectively” here means that there would be no collisions at all once the new collision avoidance scheme is applied.

Based on our previous work, in the next step, a detailed study will take place. Firstly, with fixed initial distribution area size, how different choices of parameters affect collisions will be studied. Secondly, given a set of parameters, in order to avoid coliision completely, the minimum size of initial deployment area is to be determined.

3) Further improvement of MMT

Though our trial scanning of Southbury data center is successful, it is a demo program and still leaves a lot of places to be improved.

Firstly, two major modifications would be done to improve user operating experience. The first one is use of a joystick instead of keyboard for remote control. The keyboard operation is very inconvenient and it is hard to memorize functions of different keys. Apparently, joystick is a more user interface friendly than keyboard in controlling robot’s movements. Operating MMT is more like playing video games, the operation will become funny and simple.

Besides, at this moment, views of the cameras are very limited and sometimes the operator has difficulty in locate robot’s position. So the second modification is to add more cameras on top of the mobile robot at different angles and construct a more user interface friendly view of the environment based on pictures of these cameras.

The third one is, in our final vision of this project, the robot can eventually locate itself without human being’s help. Given a layout of the data center, it should be able to planning a routine, by following which the MMT could scan the whole data center in the shortest time with least power consumption. Besides, the robot is able to detect errors and inaccuracy of the old map and correct them automatically.

Lastly, as we mentioned in Chapter 4, multi-robot system is more capable than single robot. As a result, for the MMT, it is possible to use several MMTs together rather than depending on single MMT in data center scanning. It is very obvious that this multi-MMT would be more effective in the scanning work.

Bibliography

[1]D. E. Manolakis, "Efficient Solution and Performance Analysis of 3-D Position Estimation by Trilateration", IEEE Transactions on Aerospace and Electronic Systems, 32(4): 1239–1248, 1996.

[2]D. E. Manolakis and M. E. Cox, "Effect in Range Difference Position Estimation due to Stations' Position Errors", IEEE Transactions on Aerospace and Electronic Systems, 34(1): 329–334, 1998.

[3]I. D. Coope, "Reliable Computation of the Points of Intersection of n Spheres in R^n", The Australian & New Zealand Industrial and Applied Mathematics Journal, 42(E): C461–C477, 2000.

[4]F. Thomas and L. Ros, "Revisiting Trilateration for Robot Localization", IEEE Transactions on Robotics, 21(1): 93–101, 2005.

[5]B. T. Fang, "Trilateration and Extension to Global Positioning System Navigation," J. Guidance, Contr., Dynam., 9(6): 715–717, 1986.

[6]A. El-Rabbany, Introduction to GPS: The Global Positioning System. Norwood, MA: Artech House, 2002.

[7]V. Ashkenazi, D. Park, and M. Dumville, "Robot Positioning and the Global Navigation Satellite System," Industrial Robot, 27: 419-426, 2000.

[8]J. Hightower, G. Borriello and R. Want, "SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength," UW CSE Technical Report, 2000.

[9]R. Want, A. Hopper, V. Falco, J. Gibbons. "The Active Badge Location System," ACM Transactions on Information Systems, 40(1): 91-102, January 1992.

[10] Andy Ward, Alan Jones, and Andy Hopper, “A New Location Technique for the Active Office,” IEEE Personal Communications, 4(5): 42-47, Oct. 1997.

[11] N. Priyantha, A. Miu, H. Balakrishnan, and S. Teller, “The Cricket Compass for Context-Aware Mobile Applications,” In Proc. 7th ACM MOBICOM Conf., 1–14, Rome, Italy, July 2001.

[12] Yu Zhou, Wenfei Liu and Peisen Huang, “Laser-activated RFID-based indoor localization system for mobile robots”, 2007 IEEE International Conference on Robotics and Automation, 4600-4605, May, 2007.

[13] Charles Cohen and Frank V. Koss, “A Comprehensive Study of Three Object Triangulation,” SPIE Conference on Mobile Robots, 1831: 95-106, May 1993.

[14] M. Betke and K. Gurvits, “Mobile Robot Localization Using Landmarks” IEEE Transaction on Robotics and Automation, 13: 51–263, Apr. 1997.

[15] Ilan Shimshoni, “On Mobile Robot Localization from Landmark Bearings,” IEEE Transaction on Robotics and Automation, 18: 971–976, Dec. 2002.

[16] K. T. Sutherland and W. B. Thompson, “Inexact Navigation,” 1993 IEEE International Conference on Robotics and Automation, Atlanta, GA, 1: 1–7, May 1993.

[17] A. J. Muñoz and J. Gonzalez, “Two-dimensional Landmark-based Position Estimation from A Single Image,” IEEE International Conference on Robotics and Automation, 3709–3714, 1998.

[18] H. Ishiguro, K. Kato, and M. Barth, “Identifying and Localizing Robots with Omnidirectional Vision Sensors,” Panoramic Vision: Sensors, Theory, and Application, New York: Springer-Verlag, 376–391, 2001.

[19] Stephen Se, D. Lowe, and J. Little, “Mobile Robot Localization and Mapping

with Uncertainty Using Scale-invariant Visual Landmarks," International Journal of Robotics Research, 21(8):735–758, 2002.

[20] Y. Zou, and K. Chakrabarty, "Sensor Deployment and Target Localization based on Virtual Forces", 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, 2003.

[21] M. Erdmann, and T. Lozano-P~rez, "On Multiple Moving Objects, Technical Report", Artificial Intelligence Laboratory, MIT, Cambridge, MA, 1986.

[22] L. Parker, "Cooperative Robotics for Multi-target Observation", Intelligent Automation and Soft Computing, 5(1): 5-19, 1999.

[23] L. Parker, "Distributed Algorithms for Multi-robot Observation of Multiple Moving Targets", Autonomous Robots, 12: 231-255, 2002.

[24] J. H. Reif, and H. Wang, "Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots", Robotics and Autonomous Systems, 27: 171-194, 1999.

[25] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile Sensor Network Deployment Using Potential Fields: A Distributed, Scalable Solution to The Area Coverage Problem", 6th International Conference on Distributed Autonomous Robotic Systems, 2002.

[26] S. Poduri, and G. S. Sukhatme, "Constrained Coverage for Mobile Sensor Networks", 2004 IEEE International Conference on Robotics and Automation, 1: 165-171, 2004.

[27] D. O. Popa, H. E. Stephanou, C. Helm, and A. C. Sanderson, "Robotic Deployment of Sensor Networks Using Potential Fields", 2004 IEEE International

Conference on Robotics and Automation, 1: 642-647, 2004.

[28] M. Ji, and M. Egerstedt, “Distributed Coordination Control of Multi Agent Systems While Preserving Connectedness”, IEEE Transactions on Robotics, 23(4): 693-703, 2007.

[29] M. Lam, and Y. Liu, “ISOGRID: An Efficient Algorithm for Coverage Enhancement in Mobile Sensor Networks”, 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1458-1463, 2006.

[30] M. Jenkin, and G. Dudek, “The Paparazzi Problem”, 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, 3: 2042-2047, 2000.

[31] Z. Butler, and D. Rus, “Event-based Motion Control for Mobile Sensor Networks”, IEEE Pervasive Computing, 2(4): 34–43, 2003.

[32] J. Tan, “A Scalable Graph Model and Coordination Algorithms for Multi-robot Systems”, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 1529-1534, 2005.

[33] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, “Coverage Control for Mobile Sensing Networks”, IEEE Transactions on Robotics and Automation, 20(2): 243–255, 2004.

[34] J. Tan, N. Xi, W. Sheng, and J. Xiao, “Modeling Multiple Robot Systems for Area Coverage and Cooperation”, 2004 IEEE International Conference on Robotics & Automation, 3: 2568-2573, 2004.

[35] Q. Jiang, “An Improved Algorithm for Coordination Control Multi-agent System based on R-limited Voronoi Partitions”, 2006 IEEE International Conference on Automation Science and Engineering, 667-671, 2006.

[36] M. Schwager, J. Slotine, and D. Rus, "Decentralized, Adaptive Control for Coverage with Networked Robots", 2007 IEEE International Conference on Robotics and Automation, 3289-3294, 2007.

[37] A. Winfield, "Distributed Sensing and Data Collection via Broken Ad Hoc Wireless Connected Networks of Mobile Robots", Distributed Autonomous Robotic Systems 4, L. Parker, G. Bekey, and J. Barhen, Eds. Springer-Verlag, 4: 273-282, 2000.

[38] W. Kerr, D. Spears, W. Spears, and D. Thayer, "Two Formal Gas Models for Multi-agent Sweeping and Obstacle Avoidance", Lecture Notes in Artificial Intelligence, 3228: 111-130, 2005.

[39] M. R. Pac, A. M. Erkmen, and I. Erkmen, "Scalable Self-deployment of Mobile Sensor Networks: A Fluid Dynamics Approach", 2006 IEEE/RSJ International Conference on Intelligent Robotics and Systems, 1445-1451, 2006.

[40] B. Jung, and G. S. Sukhatme, "Tracking Targets Using Multiple Robots: The Effect of Environment Occlusion", Autonomous Robots, 13: 191-205, 2002.

[41] Y. Zhou, and J. Tan, "Deployment of Multi-robot Systems under The Nonholonomic Constraint", IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 389-394, 2008.

[42] P. V. Sander, D. Peleshchuk, B. J. Grosz, "A Scalable, Distributed Algorithm for Efficient Task Allocation", Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, 1191-1198, 2002.

[43] M. I. Shamos F. Preparata, Computational Geometry: An Introduction, Springer-Verlag, New York, 1985.

[44] H. Goldstein, Classical Mechanics, 2nd edition, Reading, MA: Addison-Wesley,

1980.

[45] Y. Zou and K. Chakrabarty, “Sensor Deployment and Target Localization Based On Virtual Forces,” INFOCOM 2003: 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, 2: 1293–1303, 2003.

[46] S. Martinez and F. Bullo, “Optimal Sensor Placement and Motion Coordination for Target Tracking,” Automatica, 42(4): 661–668, 2006.

[47] B. E. Bishop, “On the Use of Capability Functions for Cooperative Objective Coverage in Robot Swarms,” 2007 IEEE International Conference on Robotics and Automation, 2306–2311, 2007.

[48] Y. Wang and Y. Tseng, “Distributed Deployment Schemes for Mobile Wireless Sensor Networks to Ensure Multilevel Coverage,” IEEE Transactions on Parallel and Distributed Systems, 19(9): 1280–1294, 2008.

[49] B. Wang, “Sensor Placement for Complete Information Coverage in Distributed Sensor Networks,” Journal of Circuits, Systems and Computers, 17(4): 627–636, 2008.

[50] J. P. Le Cadre and G. Souris, “Searching Tracks,” IEEE Transactions on Aerospace and Electronic Systems, 36(4): 1149–1166, 2000.

[51] Y. Zou and K. Chakrabarty, “Uncertainty-aware and Coverage-oriented Deployment for Sensor Networks,” Journal of Parallel and Distributed Computing, 64(7): 788–798, 2004.

[52] C. Wu, K. Lee, and Y. Chung, “A Delaunay Triangulation based Method for Wireless Sensor Network Deployment,” Computer Communications, 30(14-15): 2744–2752, 2007.

[53] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan, and K. K. Saluja, “Sensor

Deployment Strategy for Detection of Targets Traversing A Region," Mobile Networks and Applications, 8(4): 453–461, 2003.

[54] H. Goldstein, Classical Mechanics, 2nd edition, Reading, MA: Addison-Wesley, 1980.

[55] Pioneer 3 Operations Manual with MobileRobots Exclusive Advanced Robot Control & Operations Softwares, MobileRobots Inc., 2006.

[56] A.J. Davison and D.W. Murray, "Mobile Robot Localization using Active Vision,", In European Conference on Computer Vision, II: 809-825, 1998.

[57] David A. Forsyth and Jean Ponce, "Computer Vision: A Modern Approach," Prentice Hall, 20–54, 2002.

[58] R. Fletcher, "Practical Methods of Optimization," John Wiley & Sons, 2nd Edition, New York, 1987.

[59] Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431, U.S. Environmental Protection Agency, ENERGY STAR Program, Aug. 2007.

[60] American Society of Heating, Refrigerating and Air Conditioning Engineers, "Thermal Guidelines for Data Processing Environments", Atlanta, GA, 2004.

[61] "Data Center Industry Leaders Reach Agreement on Guiding Principles for Energy Efficiency Metrics", http://www.energystar.gov/ia/partners/prod_development/downloads/DataCenters_AgreementGuidingPrinciples.pdf, Feb.1, 2010.

[62] R. Hughes, "Data centers of the future", http://www.datacenterjournal.com/News/Article.asp?article_id=319, May 24, 2005.

[63] H. F. Hamann, T. van Kessel, M. Iyengar, J. Chung, W. Hirt, M. Schappert, A. Claassen, J. Cook, W. Min, Y. Amemiya, V. López, "Uncovering Energy Efficiency

Opportunities in Data Centers", IBM Journal of Research and Development, 53(3):10:1-10:12, 2009.

[64] H. F. Hamann, M. Schappert, M. Iyengar, T. van Kessel, A. Claassen. "Methods and Techniques for Measuring and Improving Data Center Best Practices", 11th Intersociety Conference on Thermomechanical Phenomena in Electronic Systems, 1146-1152, 2008.

[65] H. F. Hamann, "A Measurement-based Method for Improving Data Center Energy Efficiency", 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 312-313, 2008.

[66] PatrolBot Operations & Technical Manual, Mobile Robots Inc.