# COOPERATIVE TARGET TRACKING WITH MULTIPLE MOBILE ROBOTS

by

Roozbeh Mottaghi

B.Sc., Sharif University of Technology, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in the School
of
Engineering Science


SIMON FRASER UNIVERSITY
Spring 2006

## APPROVAL

**Name:** Roozbeh Mottaghi

**Degree:** Master of Applied Science

**Title of thesis:** Cooperative Target Tracking With Multiple Mobile Robots

**Examining Committee:** Dr. Dong In Kim
Chair

---

Dr. Kamal Gupta, Professor
School of Engineering Science
Senior Supervisor

---

Dr. Richard Vaughan, Assistant Professor
School of Computing Science
Supervisor

---

Dr. Gregory Mori, Assistant Professor
School of Computing Science
SFU Examiner

**Date Approved:** _____

# Abstract

A method for a team of mobile robots to cooperatively track a moving target has been described. We address the main limitation of existing approaches which can not cope with cases in which the target is not seen for long periods of time. A particle representation method represents the multi-modal uncertainty in the estimated pose of the target. Then a potential field guides the robots to visit as many particles as they can to reduce the uncertainty in the environment and to prevent the uncertainty area from further growing. It is shown how this method can be used as a coordination strategy whereby a team of robots cooperatively minimize the uncertainty in the pose of a tracked target. Simulation results show there is a significant difference between coordinated and non-coordinated target tracking. The method has also been tested on real robots.

**Keywords:** Target Tracking, Multiple Robots, Particle Filtering, Potential Field, Uncertainty Minimization.

*To my beloved parents*

# Acknowledgments

I am really grateful to Dr. Kamal Gupta who trusted me and gave me the opportunity to continue my studies at SFU.

Very special thanks to my supervisor, Dr. Richard Vaughan. I will never forget his generous support during my masters studies. He has been a great teacher, mentor and advisor for me. I would also like to thank Dr. Greg Mori for examining my thesis despite his busy schedule.

I wish to thank my colleagues at Autonomy lab specially Pooya Karimian for helping me during the experiments and Sarah Brown for her helpful comments for data analysis.

Some parts of this thesis are reprinted, with permission, from references number 24 and 25, © 2005-2006 IEEE.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The problem of estimating the positions of moving objects has become an important problem in mobile robotics. Using mobile robots as tracking devices has a lot of advantages. First, a mobile robot covers a larger area compared to stationary sensors so the number of required sensors will be decreased. Second, since the robot is able to move, repositioning of the robot has a direct impact on the performance and the task of target tracking can be performed more efficiently. In addition, increasing number of robots improves the performance because a larger area will be covered by the robots and information exchange among the robots would help them to track the object of interest more robustly. Nevertheless, a coordination strategy is required to prevent interference and perform the task more efficiently.

Traditional target tracking approaches suffer from failure in situations in which the targets go out of field of view of the sensors for a long periods of time. Consider the task of a mobile robot tracking a person through a building. Figure 1.1 shows the situation where a robot has followed its target down a corridor to a T-junction, and the target has left the robot's sensor field of view. Assume that the robot could not detect whether the person moved to the right or the left. So the tracker fails and the uncertainty increases over time.

The main focus of this thesis is on proposing a coordination method for mobile robots to solve this problem of existing trackers that is minimizing uncertainty of the position of the target in case of pervasive occlusion.

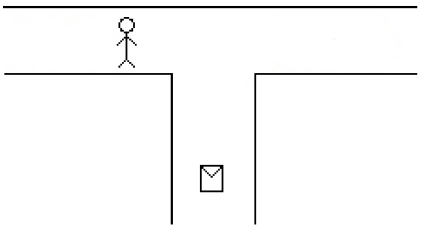


Figure 1.1: A robot (bottom) tracking a person (stick figure) who has disappeared from view.

## 1.1 Background

Probabilistic approaches to automatic target tracking have recently become popular due to their robustness in tracking in the presence of uncertainty. In particular, multi-modal representations have proved successful when targets are partially or briefly occluded [15] [14].

In addition to extensive work on target tracking in the computer vision community, several authors have described tracking systems using autonomous mobile robots. These systems can be divided into four categories:

**Single robot tracking a single target:** Most of the early approaches in target tracking with robots dealt with tracking a single object. These days, the systems are designed to track multiple objects due to demanding applications which will be discussed later. Papanikolopoulos et al. presented an early approach for tracking an object by a robot arm which is an example of tracking a single object by a single robot [26].

**Single robot tracking multiple targets:** As mentioned before, tracking multiple objects has become more popular due to the needs in various applications. As an example of recent work Schulz et al. [30] have introduced sample-based joint probability data association filters to track multiple moving objects. To keep track of multiple objects, a joint probability distribution should estimate the state of the objects but the size of state space grows exponentially when the number of tracked objects increases and makes tracking intractable. One solution is to consider one filter per target but the problem of this approach is that it will not be possible to determine which measurement is caused by which target. They apply an almost efficient Joint Probabilistic Data Association Filter (JPDAF) for this purpose.

Montemerlo et al. [22] present a probabilistic algorithm called the conditional particle

filter to track a large distribution of people locations conditioned upon the robot poses. They presented a probabilistic algorithm to simultaneously localize the robots and track the nearby people. They implemented a joint particle filter for simultaneous localization and tracking where a set of particles represented the people location and they were conditioned upon a smaller set of particles representing robot poses. These approaches are based on the decisions of a single robot which results in non-robustness against the failure of that robot.

**Multiple robots tracking a single target:** Target tracking can be improved by deploying multiple robots but a strategy is needed for coordination of the robots to improve the efficiency of tracking. Mazo et al. propose a method for formation of a team of robots equipped with short range sensors to best track a single moving object [21]. Another example has been implemented by Matsuoka et al. where multiple cameras are used to track and localize a helicopter robot [20]. The proposed method in this thesis falls into this category while it is extendable to multiple target problem. Our implemented system not only tracks the target but also coordinates the robots to find the target more efficiently if it goes out of field of view of robots.

**Multiple robots tracking multiple targets:** An extended version of the previously mentioned methods is to track multiple targets with multiple robots. Jung and Sukhatme [13] have proposed an example cooperative system for tracking multiple targets with multiple robots. They proposed a region-based approach for multi-target tracking in an indoor environment. They track the targets by assigning a region to each robot. After constructing a topological map of the environment, each robot is assigned to a region according to target density in that region. Their approach, in common with the other approaches mentioned above, works well when the target lies in the sensors' field of view or it has a short-term occlusion but unlike this thesis, they do not address the long-term occlusions which cause large uncertainty for the tracker.

Pursuit-evasion games also model a searcher chasing an evader. This class of problems guarantees that even in the worst cases in which the evaders move arbitrarily fast, any evader would be found by a group of pursuers. LaValle et al. [17] proposed a pursuit-evasion method for $k$-searchers where each searcher is equipped with $k$ flashlights by which they can see the environment. Gerkey et al. [9] recently showed a generalization of this method by introducing a new class of searcher, the $\phi$-searcher where each pursuer has a $\phi$ radian field of view.

A practical deficiency of the known solutions to the pursuit-evader problem is that they

are highly computationally intensive and do not scale well in application to multiple-robot systems. For example, in Gerkey's approach, the joint information and action space grows exponentially in the number of searchers. A further limitation is that these methods do not address the continuous tracking of intruders once they are found.

In this thesis, a new tracking and coordination technique which requires a minimal amount of communication among the agents to coordinate multiple robots is proposed in order to minimize the uncertainty about the location of a moving intruder. As mentioned above, existing tracking methods (excluding the pursuit-evasion methods) focus on combining sensor measurements to track the object of interest. The problem arises when the object of interest goes out of the field of view for a long period. Our method simultaneously addresses this problem and the problem of coordinating multiple robot trackers.

Consider the previously mentioned problem of the robot and a person at a T-junction. If the robot has a probabilistic model of the target's future movements and we represent the location of the target by a probability distribution over the possible locations in the building, two modes of the probability distribution over target location can be seen (see Figure 1.2). We represent this probability distribution by a set of particles (Particle Filtering method [12] [4] [1]). If a particle falls within the sensor's field of view, but no target is detected, the particle can be eliminated. Considering this model, we can state a simple rule that maximizes the probability of observing the target: the robot must try to observe as many particles as possible.

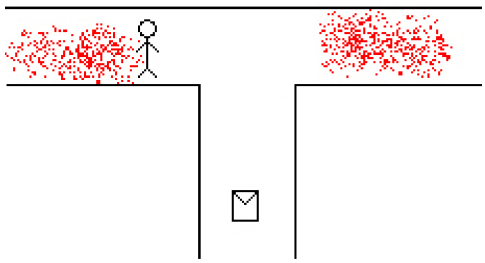


Figure 1.2: The point-cloud represents a set of hypotheses about the person's current position, generated by a probabilistic model of his movements [25]. Printed by permission of IEEE ©.

The task of our robot controller therefore is to minimize the uncertainty by maximizing the number of visited particles (the particles that lie in sensors' field of view). This approach

was taken by Rosencrantz et al. [28] to locate the opponents in a laser tag game in which the opponents might be under pervasive occlusions. However, their work mostly addressed the improvement of the tracker for multiple opponent tracking, rather than coordination of multiple trackers.

In this thesis, a particle filtering method has been implemented to represent arbitrary multi-modal densities for the location of the intruder. Then we apply a potential field method on top of the particle filtering for coordinating multiple agents to reduce the uncertainty in the environment. Each agent decreases the uncertainty in target position estimate by sweeping as many particles as it can. Coordination between agents is achieved by each robot selecting a subset of the particles to observe.

As a formal statement of the problem, the task of each robot is to minimize the entropy [32] of the particles that show an estimation for the position of the intruder. A minimum entropy is equivalent to having a dense cloud of particles which is a good estimation for the location of the intruder. Therefore, the searchers cooperate to prevent increase of the entropy.

## 1.2 Thesis Outline

Chapter two presents an outline of robot localization and mapping the environment, the primary prerequisites for a mobile robot working in an indoor environment. A detailed and intuitive description of the probabilistic tracking method is presented in Chapter 3. Chapter 4 introduces a novel technique where a particle cloud and map are combined to create a potential field robot controller for a single robot. In Chapter 5, cooperative action selection and optimization strategies for searching the environment by multiple robots are discussed. An experimental section shows the result of the implementation of the proposed method on real robots and simulated robots in different environments. Additionally, the performance of the system has been studied with respect to different population sizes, with and without communication, and also different types of communication between teammate robots. Finally, Chapter 7 discusses the advantages and deficiencies of the proposed method and some future research directions are described. Figure 1.3 shows a sequence of running the robot controller.

0. **Mapping:** An offline procedure for building a map of the environment. The inputs are laser scans and odometry and the output is a grid representing the map (*Chapter 2*).

Iterate:

1. **Localization:** An estimate of the position of the robot will be found. The inputs are the map, odometry and laser scan data and the output is the location of the robot (*Chapter 2*).
2. **Communication:** The robot sends its pose and observation to teammates and receives their information.
3. **Particle Filtering:** It is used to find an estimation for the location of the object of interest. The inputs are robot and teammates locations and observations and the output is a set of particles representing an estimation for the position of the target (*Chapter 3*). The procedure is as follows:
   (a) Resampling
   (b) Applying Motion Model
   (c) Reweighting:
      i. Sensor Model Reweighting
      ii. Checking Particle Visibility
      iii. Checking Obstacle Crossings
4. **Finding Map Distances:** The inputs are the map and the locations of the robot and teammates. The output is the map distance for each robot (*Chapter 4*).
5. **Force Calculations:** The inputs are the map, locations of the robot and teammates, the set of particles and the map distances. The output is a force that determines the next direction of the movement of the robot (*Chapters 4 and 5*).
6. **Navigation Target Selection:** The next location of the robot will be determined. The input is the computed force and the output is a cell on the map.
7. **VFH:** It chooses a path to the next location and avoids the obstacles. The input is a cell on the map and the output is a speed command for the wheels of the robot (*Appendix One*).

Figure 1.3: A sequence of running the program.

# Chapter 2

# Robot Localization and Mapping

Usually, the first prerequisite for a mobile robot working in an indoor environment is to have a representation of that environment. There are systems which do not have any global representation and decide based on local sensor information but they usually suffer from redundancies, for instance performing repeated work because of no global view. After acquiring sufficient knowledge about the environment, the robot may be able to estimate its pose (position and orientation) relative to the environment according to sensor measurements. These two steps can be performed together and the whole process is commonly referred to as Simultaneous Localization and Mapping (SLAM) or Concurrent Mapping and Localization (CML). Separation of the mapping stage from localization can somewhat help us to reduce the computational burden of the system. So the map will be acquired automatically offline, then the robot uses that map to obtain an estimate of its pose.

In this chapter, we will review the mapping problem and the current approaches to this problem. Also, the offline mapping method is described and some experimental results are shown. Then, we will describe Monte Carlo localization, a method which is widely used in recent years by roboticists in mobile robot experiments.

## 2.1 Mapping

The problem of robotic mapping is that of acquiring a spatial model of a robot's environment through robots' sensors. To build a map, robots must have sensors to perceive the surrounding world. Cameras, range finders such as laser scanners and sonar sensors, infrared sensors and GPS are the common sensors used for the task of mapping. For the experiments

of this thesis, the robots are equipped with laser scanners to collect range information of the workspace. In addition, the motion commands convey some information about the map, since they specify the location of a sensor measurement. But the motion commands are also erroneous and will not be sufficient for building an almost accurate map. The key challenges in robotic mapping are described in [31]:

- A key challenge in robotic mapping is the noise in sensor measurements. The noises are not statistically independent and a false measurement in the past will affect a future observation. Figure 2.1 shows the almost raw data gathered at SFU TASC building. As you see in the picture a small rotational error have resulted in a large error in the other corridors (the correct map is shown in Figure 2.2).

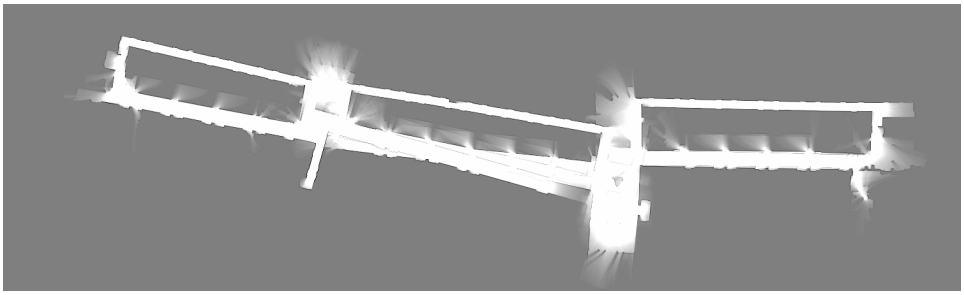


Figure 2.1: A small rotational error has caused the mis-alignments on the map.

- The other problem of robotic mapping is the high dimensionality of data which means even for representing a map of a simple environment a large amount of data should be stored. Although we represent the map by 2 or 3 dimensions, the dimensionality of the underlying estimation device is high. Current approaches try to represent a map by high-level entities instead of a detailed representation [18].

- A third problem in robotic mapping is the correspondence problem, also known as the data association problem. The correspondence problem is the problem of determining if sensor measurements taken at different points in time correspond to the same physical object in the world. For example, when a robot maps a large cyclic environment and arrives to the starting point, it should determine if this point has been visited and mapped already or it is a new area in front of the robot.

- The fourth problem is that mapping a dynamic environment would be very challenging because the sensor measurements will be inconsistent. For instance, when a robot arrives at a closed door which was previously open, it does not know whether it has arrived at another location or it is the same location with different sensor measurements.

- Finally, the robots need an exploration algorithm to move in the environment and gather the information. Because of some limitations such as risk of falling of the robot from stairs or blocking the robot by people, in this thesis, this step of the experiments have been performed using a joystick to manually drive the robot around the building.

Most of the state-of-the-art methods in robotic mapping are based on probabilistic inference. The reason for the popularity of these methods is the uncertainty and sensor noise as the key features of this problem. Thrun [31] describes different methods used in mapping such as Kalman Filtering, Expectation Maximization and Hybrid methods (combination of Kalman Filers and EM). As a recent method, Limketkai et al. introduce relational object maps in [18], an approach to building metric maps that represent individual objects such as doors or walls. Object-based representations have much higher expressive power in that they combine a high-level representation with metric accuracy.

As mentioned before, we used an offline mapping process for the real robot experiments. A brief description of the mapping software is represented in the next section.

### 2.1.1 Mapping Software

For mapping the robots' workspace, a Pioneer 3DX robot which was equipped with laser scanner was driven inside the environment and stored odometry and laser data. The map is built by using Pmap [11] which is an Open Source software and released as part of Player/Stage [10]. Pmap provides a number of libraries and utilities for laser-based mapping in 2D environments. The library maintains a Particle Filter over possible maps which means each particle (sample) represents a complete map. The output maps are topologically correct, but a little rough around the edges (some minor scan mis-alignments). According to the odometry new samples are generated (it should be noted that the odometry itself has some errors and some pre-processing on the raw odometry data is required). Then the algorithm computes the error for each point in the map according to the current sensor measurements and their correlation with the previous data. The calculated error determines

the weight of each sample. The resampling step will generate a new set of samples and the same procedure continues from the beginning. Bayes rules and Particle Filtering method will be explained in more detail in the next chapters.

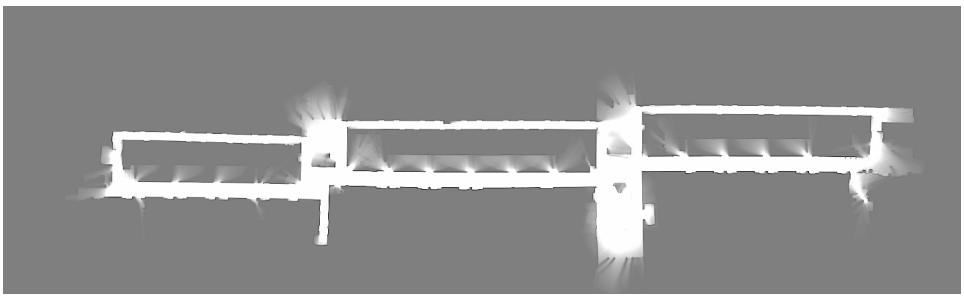


Figure 2.2: SFU TASC I building map generated by Pmap software. The white region is the free space and the grey region is the unexplored area. The represented map contains $1500 \times 450$ cells where each cell is a $9cm \times 9cm$ square.

The output of the algorithm is a bitmap image which shows the obstacles, free spaces and unexplored areas. The image is transformed to an occupancy grid map [23] in which each cell stores the vacancy status (occupied, unoccupied or unknown) of the corresponding point of the map. From now on, we mean an occupancy grid when we refer to a map.

This algorithm has some caveats. First, the space of possible maps is very large and the set of selected particles only represents sparse samples of that. Second, since each particle represents a complete map and for obtaining good results, a large number of particles are required, the algorithm would be very memory intensive and exceeding the physical memory of the host computer will slow down the process. Figure 2.2 shows a map of the 8000 level of the TASC I building at SFU generated by the pmap software using 600 samples. We used part of this area for real robot experiments described in the result section. Some refinement is done on the result of particle filter-based mapping using the libraries provided. As glass partially transmits the infrared laser beam and partially reflects it and the corridors in that level of the building are surrounded by glass, some fluctuations are visible (i.e. corridors at the bottom portion of the map).

## 2.2 Localization

So far a map is generated for the robot by the previously mentioned method. A robot should know its pose according to the provided map. The robot does not obey control commands exactly as it receives them from the controller because there are some sources of error like skidding or deficiencies in the actuators of the robot.

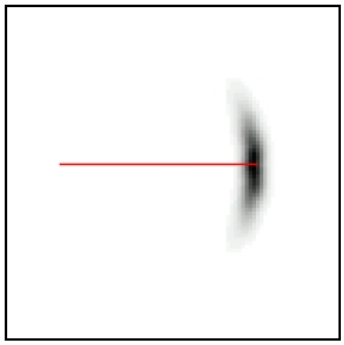


Figure 2.3: Uncertainty in position of the robot after executing the control command for going a straight line. Darker areas are related to high probability areas [33]. Printed by permission of Sebastian Thrun ©.

As you see in Figure 2.3 the robot may end up at one of the locations in the shaded area instead of going on a straight line with more probability on the darker area. So knowing only the control commands will not help us determining the position of the robot and the position should be revised according to some sensor measurements. But the sensors themselves are noisy and unreliable. So a method is required to accommodate the noise of sensors and actuators to have a good approximation of the robot's location. This problem is known as robot localization. There is extensive literature on this topic using different methods. In this section, Monte Carlo Localization(MCL) [33], a probabilistic method of estimating a robot pose is described. MCL solves the global localization and kidnapped robot problem (moving the robot from one place to another manually) in a highly robust and almost efficient way where it represents the position of a robot by a set of samples (also called particles).

### 2.2.1 Monte Carlo Localization

The key idea of Monte Carlo Localization is to estimate a probability density over the state space conditioned on the data. The following derivations and the notations are taken from [33]. The posterior is typically called the belief and is denoted by:

$$Bel(x_t) = p(x_t|d_{0..t}) \tag{2.1}$$

$x_t$ denotes the state of the robot at time $t$. The state usually comprised of 2D coordinates and the orientation of the robot. Also $d_{0..t}$ is the data available to the robot from time zero up to time $t$. The data include two types of data: odometry data which is related to motion of the robot and perception data which is related to the measurements from a sensor like a laser scanner. We can rewrite this equation with $o$ representing *observation* and $a$ representing *action*:

$$Bel(x_t) = p(x_t|o_t, a_{t-1}, o_{t-1}, a_{t-2}, ..., o_0) \tag{2.2}$$

The above equation can be transformed to the following equation by using Bayes rules:

$$Bel(x_t) = \frac{p(o_t|x_t, a_{t-1}, ..., o_0)p(x_t|a_{t-1}, ..., o_0)}{p(o_t|a_{t-1}, ..., o_0)} \tag{2.3}$$

Because the denominator is a constant relative to the variable $x_t$, Bayes rule is usually written as:

$$Bel(x_t) = \eta p(o_t|x_t, a_{t-1}, ..., o_0)p(x_t|a_{t-1}, ..., o_0) \tag{2.4}$$

where $\eta$ is the normalization constant and is equal to:

$$\eta = p(o_t|a_{t-1}, ..., o_0)^{-1} \tag{2.5}$$

Bayes filters assume that the environment is *Markov* which means the past and future states are independent given the current state, and the measurements are dependent only on the current state. Therefore, the *Markov assumption* implies:

$$p(o_t|x_t, a_{t-1}, ..., o_0) = p(o_t|x_t) \tag{2.6}$$

and hence the target expression 2.3 can be simplified to:

$$Bel(x_t) = \eta p(o_t|x_t)p(x_t|a_{t-1}, ..., o_0) \tag{2.7}$$

We will now expand the rightmost term by applying the Chapman-Kolmogorov equation[1] and integrating over the state at time $t-1$ ($x_{t-1}$):

$$Bel(x_t) = \eta p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1}, ..., o_0) p(x_{t-1}|a_{t-1}, ..., o_0) dx_{t-1} \tag{2.8}$$

Again, $p(x_t|x_{t-1}, a_{t-1}, ..., o_0)$ can be simplified by considering Markov assumption:

$$p(x_t|x_{t-1}, a_{t-1}, ..., o_0) = p(x_t|x_{t-1}, a_{t-1}) \tag{2.9}$$

which gives us:

$$Bel(x_t) = \eta p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1}) p(x_{t-1}|a_{t-1}, ..., o_0) dx_{t-1} \tag{2.10}$$

Since $p(x_{t-1}|a_{t-1}, ..., o_0)$ is the belief for time $t-1$, $Bel(x_t)$, equation 2.10 can be simplified again. The resulting equation is the recursive update equation in Bayes filters.

$$Bel(x_t) = \eta p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1} \tag{2.11}$$

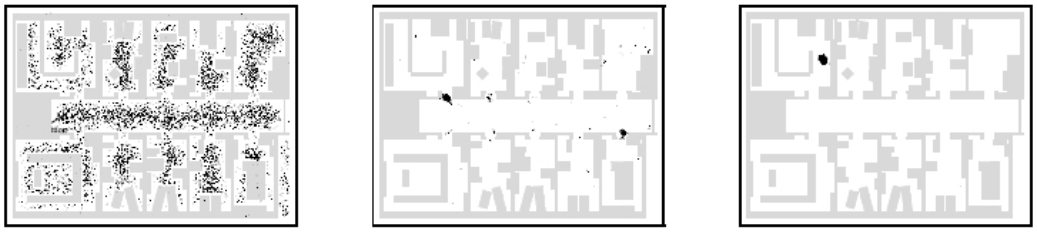


Figure 2.4: Result of Monte Carlo Localization. Successful global localization finds the true position of the robot in the rightmost picture [33]. Printed by permission of Sebastian Thrun ©.

For implementation of the above equation, we need two distributions $p(o_t|x_t)$ and $p(x_t|x_{t-1}, a_{t-1})$ which are known as the sensor model and motion model respectively. The belief at the beginning of the computations can be a uniform distribution over the possible locations. Computing the above integral is not trivial for a continuous space like the state of robot in a given map. To solve this problem, a sampling-based method (particle filtering) is used

[1] $f(X_n|X_s) = \int_{-\infty}^{+\infty} f(X_n|X_r) f(X_r|X_s)\, dx_r \quad n > r > s$

where each sample shows the location of the robot with a certain probability. The details and implementation of this method are described in the next chapter.

We use Adaptive MCL [7] function of Player/Stage [10] to perform the mobile robot localization. Adaptive MCL is referred to one case of MCL in which the number of samples changes over time and is dependent on divergence of the samples. For instance, at the beginning where the samples are uniformly distributed a large sample set is required but after some observations, this number can be reduced for the sake of efficiency. Figure 2.4 shows a result of this algorithm.

# Chapter 3

# Particle Filtering for Tracking

This chapter is devoted to describing the probabilistic method implemented to estimate and predict location of the target at any time during tracking. There is an extensive literature on target tracking in the vision community and robotics research. We adopt particle filtering (also known as the Condensation Algorithm in the vision community) as a method which is able to keep a probability distribution of the target location during tracking.

For the purpose of detecting the objects of interest (which can be a human, another robot, etc.), we use a Fiducial-finder, modeling a feature detector on a laser scanner e.g. finding a laser reflective material or reading a barcode using the laser scans. We can get the relative distance and bearing of the target by using this sensor. The relative data can be transformed to global map coordinates by simple transformations. We used particle filters as the tracker method for the following reasons:

1. Unlike the other methods like Kalman Filters, Particle Filters can accommodate arbitrary sensor characteristics, motion dynamics and noise distributions. As mentioned before we should represent multi-modal distributions in the cases which was shown in Figure 1.2.

2. Particle filters can be adapted to available computational resources by changing the number of samples.

Despite the above advantages, Particle Filters have some pitfalls. First, they are not computationally efficient compared to Kalman Filters and other similar trackers. Second, if the number of samples is too small, there is a chance that the tracker loses track of the

object. Third and the most important factor which is the main focus of this thesis is that like all of the tracking methods, this method will fail if it does not observe the target for a long period of time. As mentioned before, we develop a system to overcome this deficiency and minimize the uncertainty caused by this problem using multiple mobile robots.

The following sections explain an intuitive description, then the formal theory behind particle filters and its specific implementation for the defined task. As stated, it is a non-trivial task to compute the integral of equation 2.11. We show how a sample-based method will solve that problem. For being consistent with the tracking literature, we denote the measurement at time $t$ by $z_t$. Therefore, all of the equations and derivations would be true in the localization section if we replace $z_t$ by $o_t$.

## 3.1 Intuitive Description of Particle Filtering [24]

The tracking application of the Condensation algorithm or Particle filtering was first introduced for visual tracking of curved objects in a cluttered environment [12]. In this overview, we explain tracking of every kind of object by defining a point representation of the object. By using this algorithm we can track position and or velocity or other kinematic parameters of an object. The parameter or parameters that we want to track form the **state** vector of the object. The goal of this algorithm is to estimate the current state vector of the object of interest by choosing some random vectors from the domain of the state vectors and assigning them a probability. These random vectors are called **samples**. For instance if we want to track the position of a point which moves on a line that has length of 10 units and is located on the interval [0,10] on the **x** axis of a coordinate frame, the domain of the tracking is that interval and the samples ($[x]$) are selected randomly from that interval. For example, the samples can be vectors [1.3], [1.8], [5.0], [6.7], [9.5] which are 5 samples that have been chosen randomly. Then we assign a probability to each sample. These probabilities are the probability of actual position of the object to be the same as the randomly selected samples. So we assign a scalar probability to each sample and these scalars form a distribution of the probabilities over the state space. The details of defining this probability distribution are mentioned in the next subsections.

In Figure 3.1 the estimation of position of an object in the image plane is shown. The quadrangle is the object of interest and each cross is a sample and the result of condensation tracker is a probability which is assigned to each sample. This probability is the probability

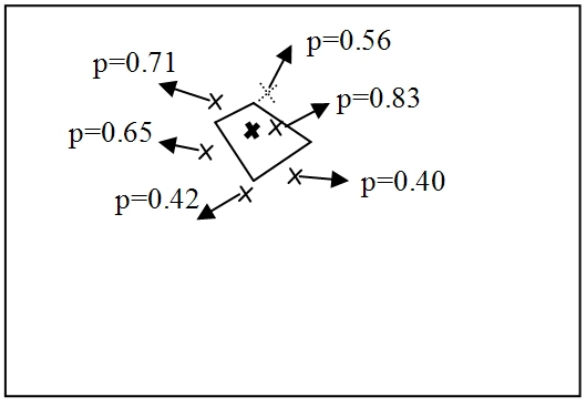



Figure 3.1: The object of interest is the polygonal object and each cross is a sample to which a probability is assigned. The thick cross is an arbitrary point of the object (we consider the object as a point). The probability that the thick cross is at the position of the dotted cross is 0.56 [24]. Printed by permission of IEEE ©.

of the presence of one specific point of the object of interest in the position of the corresponding sample. For example the probability that the thick cross in Figure 3.1 (consider the object as a point) be at the position of the dotted cross is 0.56. It should be noted that the value of the components of the samples and the number of samples may change. As the environment changes, we choose new samples and assign new probabilities to them by using the dynamic model of movement of the object and an observation made by sensors. Usually the dynamic model of the object that we want to track is not known and we guess a model for that. It is also possible to learn the dynamic model from the previously known data from the movement of the object.

In this thesis, the observation is usually the result of processing of the laser scans and finding the position of the object on the map. The theory behind the algorithm including the details of choosing samples and assigning the probabilities and finding the probability distributions, are described in the next subsection.

## 3.2 Formal Theory

The general idea of the Condensation algorithm is to find a probability distribution which approximates a probability function for each sample of the state vector of the object according to the real measurements (observation) from a sensor. The state vector at time $t$ which is denoted by $x_t$ is a vector of variables that we want to estimate (depending on the application).

The measurement from the sensor (the fiducial finder in our tracking case) at time $t$ is denoted by $z_t$. So far we have three terminologies: state, sample and measurement. The state vector consists of the variables that we want to estimate and we refer to state space as the space in which those variables (position, velocity, etc.) can change. Sample vectors are specific vectors in the state space which have been chosen randomly according to a probability distribution. A measurement is a vector of the same form as the state vector and the values of its parameters are read from the sensors. Since the sensors are noisy or have a limited range we can not absolutely rely on the sensor data (measurement) and we find a probability for the similarity of the guessed samples with the real status of the object. In our problem, we try to estimate the position and orientation of the target on the map. So, we define $x_t = [x, y, \theta]$ where $x$ and $y$ are 2D Cartesian coordinates of the object on the map and $\theta$ represents its orientation. We find a probability distribution for the state space according to the measurement, $p(x_t|z_1, z_2, ..., z_t)$, that is the probability that the state at time $t$ is equal to $x_t$ provided that the measurements from time 1 up to time $t$ are equal to $z_1$,$z_2$,...,$z_t$ respectively. Using Bayes' rule, ($p(A|B) = p(B|A)p(A)/p(B)$, $p(A|B) = p(A \cap B)/p(B)$), $p(x_t|z_1, z_2, ..., z_t)$ is computed as follows:

$$p(x_t|z_1, ..., z_{t-1}, z_t) = \frac{p(x_t|z_1, ..., z_{t-1})p(z_t|x_t, z_1, ..., z_{t-1})}{p(z_t|z_1, ..., z_{t-1})} \tag{3.1}$$

Since the measurement at time $t$ is independent of the previous measurements given $x_t$, according to the above rules $p(z_t|x_t, z_1, ..., z_{t-1}) = p(z_t|x_t)$. Also $p(z_t|z_1, ..., z_{t-1})$ is a constant. Therefore:

$$p(x_t|z_1, ..., z_t) = kp(z_t|x_t)p(x_t|z_1, ..., z_{t-1}) \tag{3.2}$$

We can compute $p(x_t|z_1, ..., z_{t-1})$ by applying the dynamic model of the object motion to $p(x_{t-1}|z_1, ..., z_{t-1})$ which is known from the previous time step. The dynamic model is an *a priori* known motion model of the object and it relates the state vector at current time step to that of previous time step and it depends on the intrinsic nature of the object and

the environment in which the object moves. It can also be computed online according to the data from tracker. The dynamic model can be defined as:

$$x_t = f(x_{t-1}) + stochastic\ \ part \tag{3.3}$$

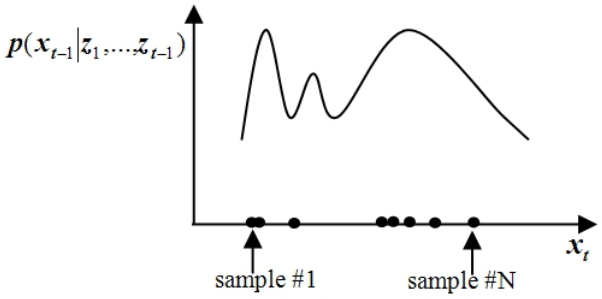
where the stochastic part can be any function for instance a vector of independent standard normal variables which are scaled by a factor that is determined according to the nature of the tracked target.

$f$ can be any function and relates the current state of the samples to the previous state. Because the movement of the object is random (i.e. unpredictable from the point of view of the tracker) we add a stochastic part to the dynamic model to add randomness to the deterministic model. For example, the intruder can stop or move backward instead of moving forward which is forced by the deterministic model. But we know that motion of a human is not totally random. As an example, consider the simple one dimensional case of tracking a point on a line where the state and measurement vectors consist of only the position of the object. As shown in Figure 3.2($a$), we draw N samples randomly according to the probability distribution from the previous time step. As you see the samples are denser in high probability areas because there is more probability that a sample is chosen in that area.

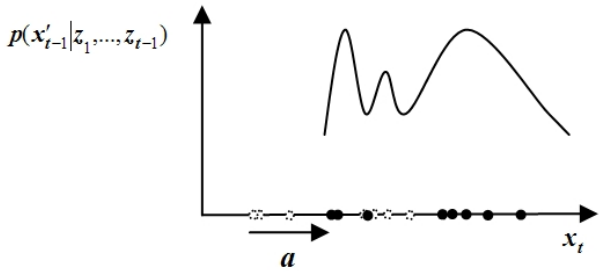
Assume that the deterministic equation of motion of the object is $x_t = x_{t-1} + \mathbf{a}$ where $\mathbf{a}$ is a constant. Figure 3.2($b$) shows the samples after applying the deterministic part of the dynamic model. Figure 3.2($c$) shows the result of applying of the whole dynamic model (deterministic + stochastic parts) to find $p(x_t|z_1, ..., z_{t-1})$.

So far, we have found $p(x_t|z_1, ..., z_{t-1})$ which is needed for computing $p(x_t|z_1, ..., z_t)$. As mentioned before we have a set of samples from the space of the state vectors. These samples were primarily drawn randomly according to the previous time step probability distribution. Then we applied the dynamic model to that set of samples to get a new set of samples. This new set of samples is the original one which has drifted by applying the dynamic model. After this step we make an observation by using the sensor and we adjust the weight (probability) of each sample according to this new observation. Three cases have been considered for the reweighting step:

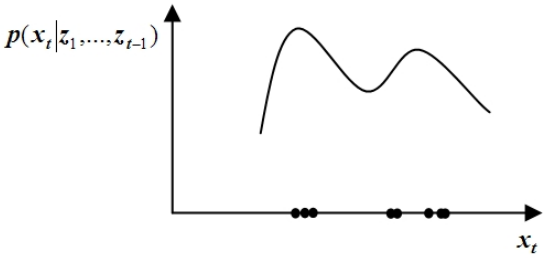
1. We define the area $S$ which is a circular segment as the sensor visibility of the agents. This area is centered at the robot position and its central angle, $\varphi$, is in the range $\theta - \frac{F}{2}$




(a) N samples are chosen randomly according to the probability distribution from the previous time step. The samples are denser near high probability areas.




(b) This figure shows the set of samples after applying the deterministic part of the dynamic model. Since the equation is $x_t = x_{t-1} + \mathbf{a}$, each sample is drifted by the size of $\mathbf{a}$. The white dots are the samples which are initially drawn and the black dots are those samples after applying the deterministic part of the dynamic model.




(c) The result of applying the stochastic part of the dynamic model to the previous set of samples is shown.

Figure 3.2: Weighing schemes [24]. Printed by permission of IEEE ©.

to $\theta + \frac{F}{2}$ where $\theta$ is the robot orientation and $F$ is the field-of-view angle subtended by the sensor. It should be noted that not having a perfect localization will not affect the method. Finally, the radius of the circular segment shows the sensor range. If the $i^{th}$ sample $s_i \in S$ and the line that connects the sample to the robot does not intersect an obstacle, $w_i$ the weight of that sample will be zero. This case is shown in Figure 3.3. Thus the robot deletes the particles that it now knows do not correspond to the real position of the target.

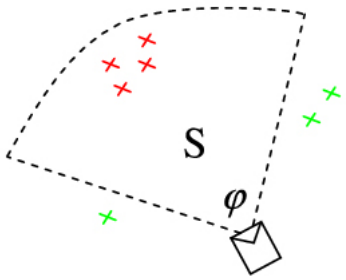



Figure 3.3: The robot assigns low weights to the visited particles [25]. Printed by permission of IEEE ©.

2. If $\overline{s_i^t s_i^{t-1}} \cap C_{obs} \neq \emptyset$, $w_i$ the weight of $i^{th}$ sample would be zero. $\overline{s_i^t s_i^{t-1}}$ is the line segment that connects the position of sample $i$ at the current time step to its position in the previous time step and $C_{obs}$ is the space of all of the obstacles present in the map. Intuitively, it means if the particles go inside an obstacle or through a wall, their weight becomes zero. Figure 3.4 shows this case. This models the constraints on the target that it can only move through free space.

3. If have we an observation of the object, the Factored Sampling method [12] is used to find the new weight of the samples. If the number of samples goes to infinity the distribution of samples from $p(z_t|x_t)p(x_t|z_1, ..., z_{t-1})$ tends to be that of $p(x_t|z_1, ..., z_{t-1}, z_t)$. A reasonable assumption for accommodating noise in the sensor model is to be a Gaussian function where the mean of the Gaussian is located on the real measurement from the sensor and its deviation is determined according to the sensor and the map. Figure 3.5 shows the reweighing of each sample according to the Gaussian

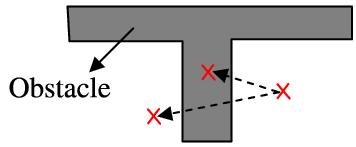



Figure 3.4: The particles which cross an obstacle will be given zero weight [25]. Printed by permission of IEEE ©.

function. It should be noted again that we have shown the observation density for a one-dimensional case. A higher degree function needed for higher dimensional state spaces. Details of the Factored Sampling Method is explained in the next subsection. In this algorithm, $p(x_t|z_1, ..., z_{t-1})$ and $p(z_t|x_t)$ have the same role as $f_1(x)$ and $f_2(x)$ in the explanation respectively.

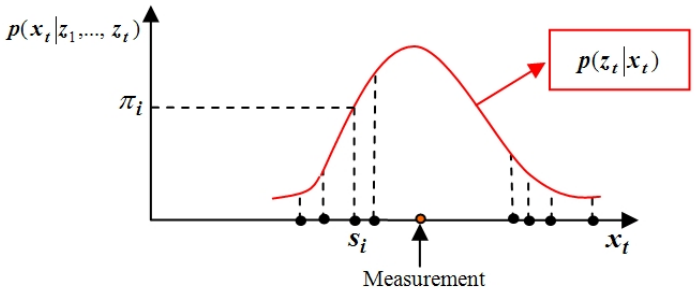



Figure 3.5: Reweighing the particles according to the sensor model [24]. Printed by permission of IEEE ©.

This set of samples with their new probabilities forms the distribution that we were looking for i.e. $p(x_t|z_1, ..., z_{t-1}, z_t)$. In the next time step $t + 1$, we use $p(x_t|z_1, ..., z_{t-1}, z_t)$ as the previous time distribution and we draw N new samples according to this distribution and we repeat the whole procedure.

If none of the above cases happened, each sample keeps its previous weight or we can assign an equal weight to all of the samples. In the traditional particle filter tracking system, some clustering method is used to decide what the current 'actual' estimate is. Usually the

particle with the mean or median of the weights is considered to be the true target pose. We avoid this clustering step, and thus avoid the artifacts it can introduce. The clustering methods can not cope with cases happened at T-junctions in the previously mentioned examples. As we may not have an observation during the tracking, we try to maximize the number of visible particles (to observe locations of more particles by the sensors) while simultaneously optimizing the joint motion of the robots.

### 3.2.1 Factored Sampling

Suppose we have a probability distribution that is the result of multiplication of two other distributions. The factored sampling method is used to find an approximation to the probability density function by using samples from those two distributions.

Assume $f(x) = f_2(x)f_1(x)$, where $f_1(x)$ and $f_2(x)$ are two probability distributions. A set of samples $s = \{s_1, s_2, ..., s_N\}$ is drawn randomly from $f_1(x)$ (Figure 3.6).

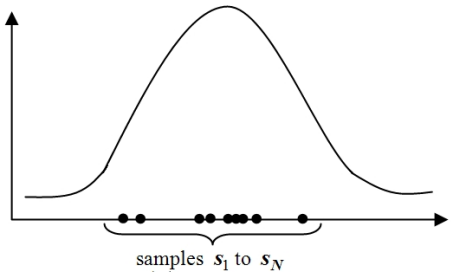
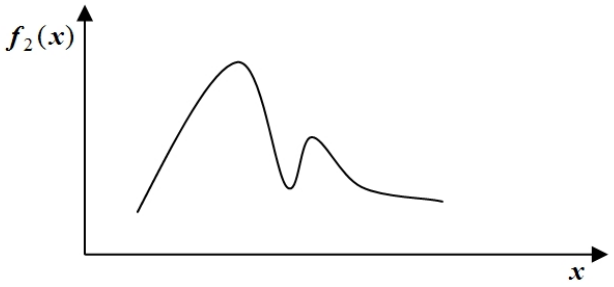



Figure 3.6: Sampling from $f_1(x)$. A set of N samples are drawn randomly with a probability proportional to $f_1(x)$. So we see more samples under high probability areas [24]. Printed by permission of IEEE ©.

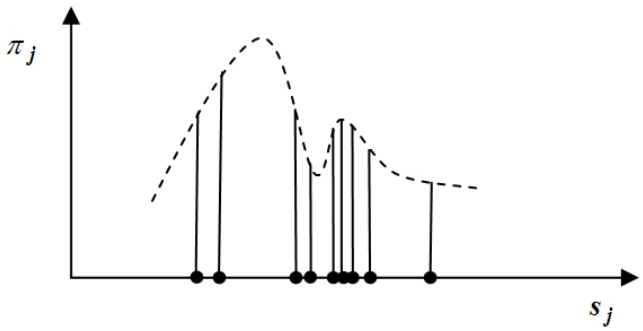
Then we find the probability assigned to each sample in proportion to $f_2(x)$. The probability $\pi_j$ of the $j^{th}$ sample in proportion to distribution of $f_2(x)$ is computed as follows:

$$\pi_j = \frac{f_2(s_j)}{\sum_1^N f_2(s_j)}, \quad j = 1, ..., N \tag{3.4}$$

So we have found a new sample set, where the distribution of the probabilities of the new samples tends to that of $f(x)$, as $N \to \infty$. Therefore, the distribution of the probability of these new samples is an approximation to the distribution $f(x)$ (Figure 3.7).




(a) $f_2(x)$ is shown in this figure.




(b) Each sample is reweighed according to equation 3.4. $\pi_j$s are the new weights.

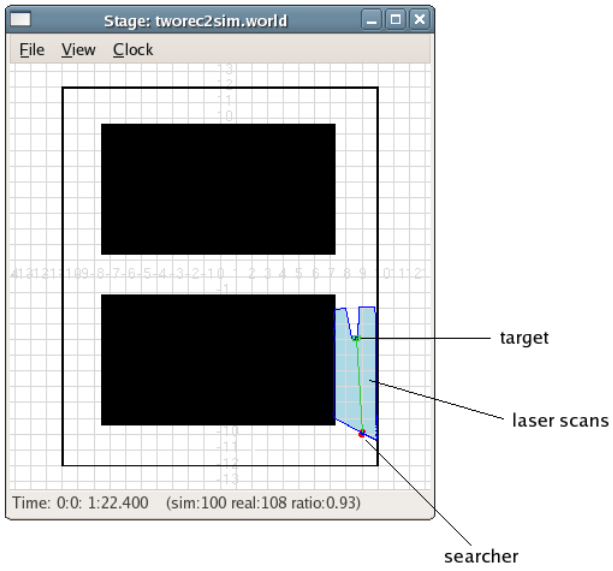
Figure 3.7: Weighing schemes [24]. Printed by permission of IEEE ©.

## 3.3 Implementation and Simulation Results of Target Tracking

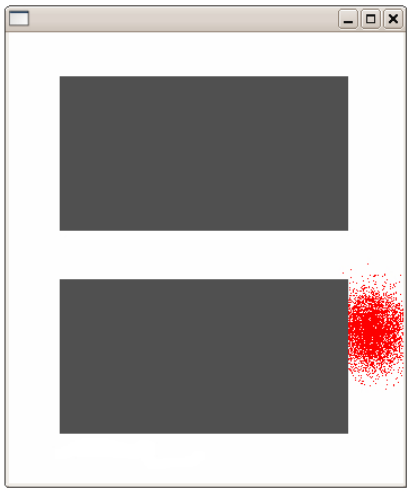
We implemented the particle filtering algorithm for tracking with the robots and the simulation results are shown in this section. A robot which is equipped with laser scanner follows another robot and the particle filter gives an estimate of the position of the target. Two cases have been considered: the target lies in the robot's sensor view and the target goes out of field of view. Some assumptions have been made for tracking: the sensor model, $p(z_t|x_t)$, is a Gaussian with $\sigma_r = 12cm$ and $\sigma_\theta = 10\ deg$ where $\sigma_r$ and $\sigma_\theta$ are deviations for range and bearing of the sensor. The other assumption is that the target obeys a first order linear motion model augmented by a Gaussian noise. The motion model can be any function but in this case a simple model is assumed.

1. Direct sensing: this case happens when the robot observes the target directly. So it updates the particles according to the sensor measurement by the Factored Sampling method. Figure 3.8 shows this case. The particle cloud shows the possible positions of the target according to target motion model.

2. No observation: in this situation the target goes out field of view of the robot and the robot does not have any observation of the target. Again, the particles show the estimated pose of the target but a multi-modal distribution might be formed. As shown in Figure 3.9, the particles are confined to the free space of the map and the particles which are in the robot's view and can be observed to not be the true position of the target have been assigned a zero weight and removed. It should be noted that the particles whose direction is the same as the last visible direction of target movement, are assigned a higher weight, so there are larger number of particles on the top portion. This models the assumption that the target continues to move in the same direction with high probability.

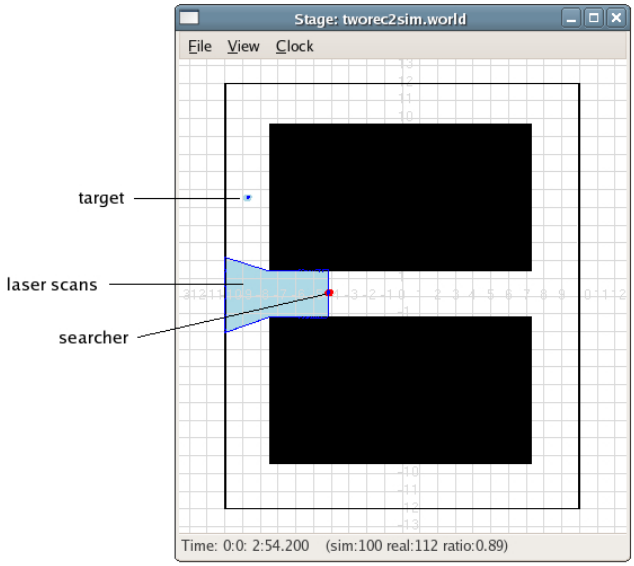



(a) Stage [34] snapshot.

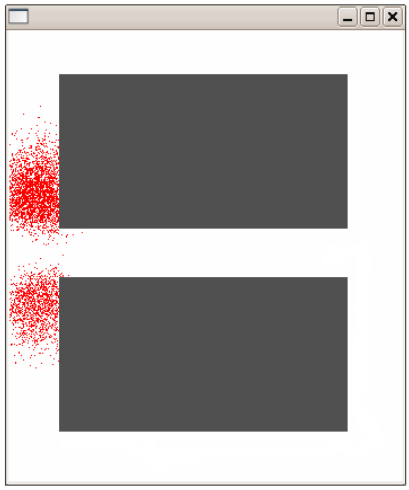


(b) Robot's view. The particle cloud shows the possible positions of the target.

Figure 3.8: Simulation results for direct sensing.




(a) Stage [34] snapshot.



(b) A bi-modal distribution shows the uncertainty in the pose of the target.

Figure 3.9: Simulation results for a robot without target observation. This is the scenario described in Figure 1.1.

# Chapter 4

# Particle Filter and Potential Field Integration

Previous chapters described robot localization and mapping and target tracking as some standard tools used by robotics community. The main contribution of this thesis begins from this chapter. Our goal is to minimize the uncertainty in target position by maximizing the number of visited particles. Also, the likelihood of finding the target is maximized if all the particles are swept off. The problem of most of the existing methods such as POMDPs (Partially Observable Markov Decision Processes) which seems well-suited to solve this kind of problem, is that their computational complexity or memory needs grow exponentially with the number of robots. POMDP is a method for decision making and optimization where the entire world is not available [29], in our case, the target position and the actions of teammates are not available to the robots.

We implement a potential field method for doing this task which is $O(N_p n)$ in the worst case where $n$ is the number of cells if we represent the map by grid cells and $N_p$ is the number particles used in the tracking algorithm. For speed, the number of particles which are used for the calculation of the forces can be decreased by selecting at random a subset of the particles. Since the particle filtering results in producing more particles in the high probability areas, the chance of choosing particles in those areas would be higher and the distribution of the particles is approximately the same. So we perform the update stage for the whole set of particles but calculate the forces based on the randomly chosen particle subset. Potential Field methods were used for motion planning in their early

robotics applications, where a robot was considered as a particle which moved in a free space. Obstacles exerted repulsive force on the robot while the goal exerted an attractive force on it and the resultant force caused the robot to move from a source to a goal without colliding with obstacles [16].

In the following subsections we explain a method for finding the distances on a map (instead of Euclidean distance) and after that, the application of this method for tracking is described.

## 4.1 Building a Traversible Map

The algorithm which is presented in this chapter performs as a motion planner and tracker simultaneously. Therefore, the result would be the shortest obstacle-free path from the robot location to the specified target. The first requirement of this method is to construct a traversible map based on the original map. The idea is the same as finding a C-space as described in [16]. As one example, Figure 4.1 shows a case where there is a free space but the robot can not go through because of its size. So, the calculations for tracking and motion

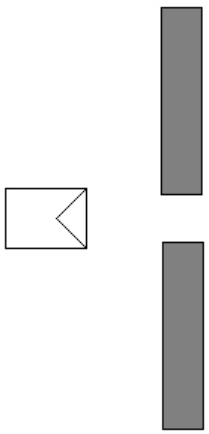


Figure 4.1: The robot does not fit into the free space between obstacles on the original map.

planning should not consider that space as free. To build a traversible map, the original map should be modified to specify this kind of free space and to prevent the planner and the tracker considering a path in this space.

The procedure for building a traversible map is as follows. We assume a certain radius for the robot and increase the size of obstacle cells by the size of the robot which means the

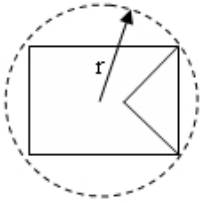



Figure 4.2: A circumcircle with radius $r$ will be assumed as the shape of the robot.

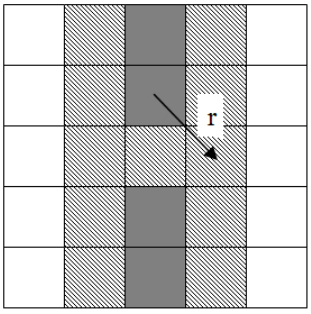



Figure 4.3: Solid gray cells are real obstacle cells. Shaded cells are marked as virtual obstacles after increasing the size of obstacles. White cells are traversible space.

cells fall into that radius around an obstacle cell, are marked as an obstacle cell. Figure 4.2 shows an approximation to the radius of that robot and Figure 4.3 shows the traversible map built for the example shown in Figure 4.1. As shown, the free space between the obstacles are filled with virtual obstacle cells, so no path will be generated through that free space but the robot can see through it.

## 4.2 Finding Map Distances

For maximizing the number of visited particles, the idea is that each particle exerts a force to the robot to attract it. So the greater number of particles in an area, the larger is the force which is imposed on the robot. This force is inversely proportional to the distance from the particle to the robot. This means that the robot tends to sweep the nearest particles first. But when the robot's mobility is limited by obstacles, as shown in Figure 4.4, the Euclidean distance from particle to robot does not indicate how quickly the robot can reach the particle. Instead we must calculate the shortest traversible path using the map.

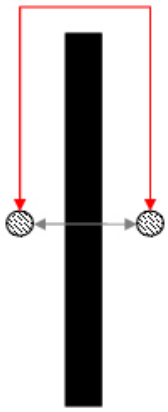


Figure 4.4: The gray arrow shows the Euclidean distance between two points which is not useful for calculating the attraction force because of the presence of the obstacle [25]. Printed by permission of IEEE ©.

Shortest-traversible-path calculations are done using a simple occupancy-grid flood fill method [16], though any equivalent method could be substituted. The distance algorithm outputs a value which is assigned to each grid cell and shows the distance of that cell from

the cell where the robot is located. The flood-fill works as follows: First, we assign a zero value to the cell in which the robot is located and an infinite number to the obstacles. Then, we pick one of the free cells around the robot cell and increment its value by one and put that cell in a queue and pick another neighbor cell until there is no cell around the current cell without an assigned value. The order of picking the neighbors is important and we pick only top, bottom, left and right neighbours. After that, we pop the first cell in the queue and perform the same procedure for its surrounding cells. This algorithm is continued until there is no cell in the queue. This method returns the minimum map distance of a point to the current position of the robot and its time complexity is $O(n)$ if it is implemented by a queue where $n$ is the number of cells on the map. Figure 4.5 shows an output of this method for measuring the map distance of a cell of the map.

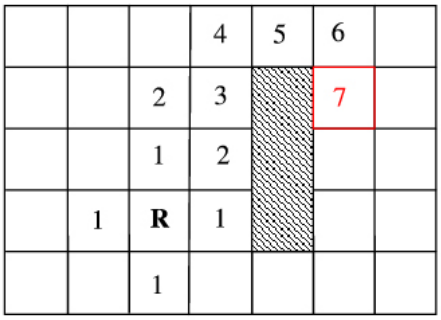



Figure 4.5: The map distance of the red cell (marked 7) from the robot cell (marked R) is calculated by using the flood fill method. The shaded area is an obstacle on the map.

Thus we find the distance of each particle from the robot as required for the calculation of the forces exerted by the particles. These calculations are explained in detail in the next subsection and in the coordination strategy section. For simplicity, from now on we represent the map distance of a cell, which is located at row $i$ and column $j$, from the robot cell by $\Delta(i, j)$.

## 4.3 Computation of Potential Forces

The navigation of our robots is based on the total force which is exerted on the robots by randomly selected particles. That means at each time step, we apply the normalized total force to the robot to find its next target position. An underlying position device based on

the extended Vector Field Histogram (VFH) [2] performs the task of avoiding local obstacles while moving according to the potential field. VFH method has been provided by Player [10] and the detailed description of it can be found in Appendix A. To compute the total potential acting on a robot, we find the force vector for each particle. Then, we sum the vectors to find the magnitude and direction of the resultant total force.

To find the approximate direction of the resultant of the total particle forces, we start from the cell where the particle is located and we check its surrounding cells, the cell with the minimum value will be selected. We continue performing the same procedure for the minimum-value cell until we reach a certain distance from the robot cell (this approximate distance is indicated by the circle in Figure 4.6). The direction of the force is approximated by the direction of the vector from the robot to the cell that is reached through the above procedure. The reason that we do not use directly the vector from the robot to the particle, is that the vector may intersect the obstacles that has blocked the robot way. Figure 4.6 shows an example of finding the force direction. The dashed line shows one of the paths from the red cell (marked 7) to the robot cell and the vector from the robot to the cell with value 3 can be considered as the force direction. Choosing this threshold value is empirical and it is better to be somehow bigger than size of robot.

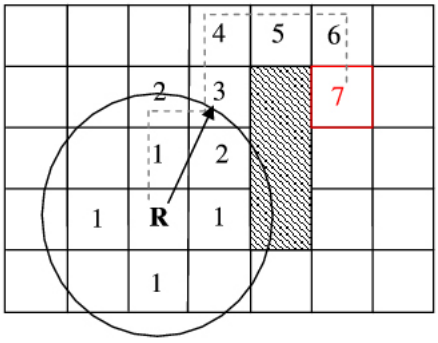



Figure 4.6: The vector shows the direction of the force which is exerted from a particle located in the red cell (marked 7) [25]. Printed by permission of IEEE ©.

If $m_i$ and $n_i$ are the row and column index of particle $i$ in the map grid, the magnitude of the force exerted by that particle, $F_i$, is calculated by the following Gaussian model:

$$F_i = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{\Delta^2(m_i,n_i)}{\sigma^2}} \tag{4.1}$$

where the $\sigma$ is assumed to be a constant in the whole process of tracking or it can be

determined according to particles data. $\sigma$ is the parameter that determines the priority for sweeping the particles close to the robot. The reason for choosing a Gaussian function is to assign more priority to the closest particles while the deviation is controllable by $\sigma$. This equation means that the closer particles exert a larger force and the first priority of the robot is to sweep the nearest particles. Nevertheless, if the number of particles is large in an area the robot will be attracted to that area neglecting the nearest particles. The magnitude and direction of the attractive force is determined by the vector summation of the forces from all of the particles which were selected randomly from the whole set of particles. The robot will be driven around according to the direction of the resulting force.

The main criticism of potential field methods in general is that rapidly changing local optima can cause an oscillatory behaviour in the navigation of the robot. However, because of the random nature of the particle filtering method and clearing of the particles during the navigation, the symmetry breaks and we have not observed adverse oscillations in the robot movements. There are a lot of tricks to get rid of the oscillation caused by potential field methods. In the next subsection, a method is proposed for removing the oscillations. This method is very easily extended to rationally coordinate multiple tracking agents, as described in the next chapter.

### 4.3.1 Preventing Local Optima

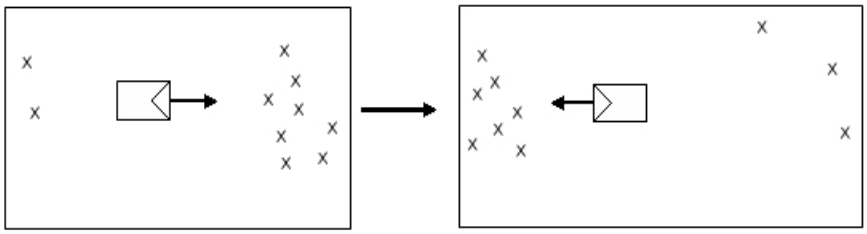


Figure 4.7: An example figure showing local optimum problem.

Figure 4.7 shows one case where a local optimum can happen. In the left picture, there are a larger number of particles in front of the robot, so the robot is attracted toward

them. After removing some of the particles which do not belong to the real position of the target, if the target is not found, the number of particles will be increased behind the robot. Therefore, the robot may leave clearing all of the particles in front and turn back to go toward the set of particles at the back. One of the methods that can help significantly in this situation is that the robot remembers the magnitude and the direction of the past attracting forces and takes them into account in the calculation of the current force. The details can be found in the next chapter. Although this method improves the performance, it is not a complete solution.

# Chapter 5

# Coordination Strategies

This chapter describes the advantages and disadvantages of multi-robot systems in general and explains why a team of robots is particularly suitable for the defined task. After that, the details of the coordination strategy of the robots for target tracking and uncertainty minimization are explained.

## 5.1 Why Use a Multi-Robot System?

Although most mobile robotics systems involve only a single robot, often there are advantages in using a group of robots which cooperate to accomplish a defined task. Better performance, increased reliability through redundancy and a reasonable cost according to the requirements of an application are the main reasons that attracted researchers toward Multi-Agent Systems. Conferences such as Distributed Autonomous Robotic Systems which are specialized for multiple robot applications, show the significant community interest. Parker [27] mentions primary research topics in this field including biological inspiration, localization, mapping, exploration, object transport and manipulation, motion coordination, and so forth. Various examples have been provided for each research topic.

A team of robots usually consists of robots of different or similar type that should communicate with each other to perform a cooperative behavior. The communication can take place via a direct channel or it can be via sensing the other robots' changes like the method described in [35]. There are some teams of robots that perform a task without any communication. Although interesting and more reliable, usually there is a lot of redundancy in these systems. Therefore, one factor that should be considered in designing a Multi-Agent

System is the inter-agent communication which has direct impact on the performance and reliability of the system. Biological and ethological studies have shown these inter-group communications among the group members in nature. Communication of the location of a goal is an instance of interaction of group members in nature [5].

A speedup will be achieved if we can find a method to break a task into components and perform them in parallel. Since each agent in a team is used as a computational resource, parallelism and distribution of intelligent components is inherent of Multi-Agent Systems which results in better performance. Although increasing the number of agents is usually useful to improve the efficiency of performing a task but the performance might be degraded after addition of certain amount of agents because of interferences. It should be noted that finding the most suitable number of agents for a specific task even in a pre-defined environment is usually a NP-Hard problem. An example case is proved by Gerkey et al. in [9] for determining the minimum number of agents for a pursuit-evasion problem in a given environment.

From the reliability perspective, Multi-Agent Systems or Multi-Robot Systems as their special case are more robust compared to a single (and often complex) robot since the failure of a single agent will not result in overall failure of the system. In addition, some tasks are ideally suited to multiple robot systems. Consider a task which has high potential for damage to an individual e.g. mine deployment or bomb disposal. Using multiple robots for these tasks obviously provides us with more reliability.

The nature of the task itself also determines the requirement of using multiple agents. For example, if two tasks are supposed to be done at the same time, it would be impossible for a single agent to do the task. On the other hand, there exist tasks which do not benefit from the use of additional agents in order to solve them. Task and environment can combine to remove any benefit of the use of multiple agents. For example, a single task at a single location does not benefit from the use of multiple robots, as a single robot is both necessary and sufficient [5]. Therefore, despite all of the advantages described above, usage of multiple robots will have obvious disadvantages for certain tasks in certain environments.

The main factors that are considered for categorizing Multi-Robot Systems are listed below. The trade offs between performance, reliability, cost, etc. of a system are caused by these parameters and a designer must specify them carefully to achieve the desired result.

- Team Size: The number of agents in a team.

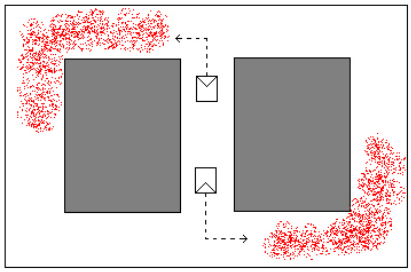
- Communication Range: The maximum distance between two members of a group such that communication is still possible.
- Communication Topology: Of the robots within the communication range, those which can be communicated with.
- Group Composition: Whether the group members are homogeneous (have the same capability)or heterogeneous (have different capabilities) [5].

## 5.2 Cooperative Uncertainty Minimization

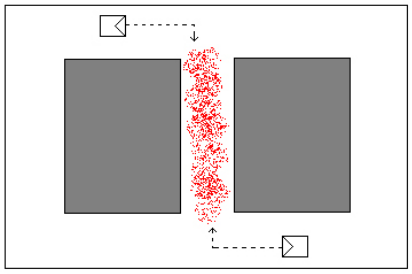
The expected uncertainty minimization and tracking performance of the system can be improved by simply adding more robots, but to maximize performance, the robots' actions should be coordinated in some way. In this section we describe how multiple robots can cooperate to perform the assigned task according to the potential fields which have been formed by the particles. The assumption for the coordination method is that each robot has an estimate of the location of the other robots. Each robot can send its global position information to the teammates through communication or it can localize the other robots in its coordinate frame. Both of these constraints are feasible using current methods and since the exact position of the other team members are not required, any other method can be taken. In our simulations, we use communication among the robots. The communication can be direct communication between two robots or in the case of limited communication range, a robot can get the location information of one robot through communication with a third robot.

As stated before, we want to minimize the uncertainty by maximizing the number of visited particles (observing the locations of more particles). So our goal is to cover an area that is occupied by larger number of particles and to prevent the particles from further spreading. Two simple cases are shown in Figure 5.1. The first figure shows the case where we have two high density regions that means the chance of finding the intruder is high in those two regions. The best action to minimize the uncertainty is that one robot goes toward one cloud of particles and the other robot goes toward the other cloud. The next figure shows the case that there is one high density area. The best action to shrink the particles' area and prevent it from further growing is that the robots approach the covered area from different directions. Our coordination method tries to achieve the above goals

while minimizing the path that a robot navigates.



(a)



(b)

Figure 5.1: Simple cases of minimizing the uncertainty by two robots [25]. Printed by permission of IEEE ©.

For the coordination of the motion of the robots, we compute the cooperative forces which are exerted by the set of particles on each robot. These forces will determine the navigation direction of the robots. First, we assign a value to each particle according to density and distance of the robots. The more negative the value, more desirable for the agent to go toward that particle. This value which is represented by $V_{n,j}$ for particle $n$ relative to agent $j$ is determined by:

$$V_{n,j} = \sum_{i=1, i\neq j}^{N} \begin{cases} -w_i F_{nj} & \Delta_i > \Delta_j; \\ w_i F_{ni} & \Delta_i \leq \Delta_j. \end{cases} \tag{5.1}$$

where $F_{nj}$ and $F_{ni}$ are the forces that particle $n$ imposes on agent $j$ and agent $i$, respectively and are computed according to Equation 4.1. $N$ is the number of agents used for tracking and $w_i$ is a priority factor which is used to assign higher priorities to some agents. Also, $\Delta_i$ and $\Delta_j$ are the map distance of the $n^{th}$ particle from agent $i$ and agent $j$. Intuitively, this equation means that the parameter $V$ will be more positive for a selected

particle and a specific robot if the density of the other robots around the particle is high. That means the robots will take care of the particles nearest to them. We normalize these values to get positive force magnitudes. The normalization is done by an exponential function again. So, the force magnitude that particle $n$ exerts to agent $j$ in presence of the other robots, $F_{n,j}$, is calculated as follows (note that $F_{n,j}$ is different from $F_{nj}$ since $F_{nj}$ is that force without the presence of the other robots):

$$F_{n,j} = e^{-(V_{n,j}-V_{min})^2} \tag{5.2}$$

where $V_{min}$ is the most negative value. The direction of the force is also found by the procedure described in the last section. Now, we should find the vector sum of the forces which are exerted to one robot by the set of particles, So:

$$\vec{F_j^{tot}} = \sum_{n=1}^{N_s} \vec{F_{n,j}} \tag{5.3}$$

where $N_s$ is the number of randomly selected particles. As mentioned before, for the sake of efficiency, we use a small set of particles for force calculations and only the update step (in particle filtering) is done for the whole set of particles. The direction of navigation of robot $j$ is dependent on $\vec{F_j^{tot}}$. In the next chapter, the simulation results and real robot experiments of this coordination method are shown.

## 5.3 Force Calculation for Avoiding Local Optima

As mentioned in the last chapter, one of the caveats of this tracking method is that rapidly changing local optima can cause an oscillatory behavior. There are some tricks to overcome this problem. One way is that the robot has memory of past forces. In this case, the previous forces should be considered in the calculation of the total force. As the figure shows there is a large cloud of particles behind the robot so the robot will stop clearing the particles in front and leave that area heading toward the larger cloud (because $F_L > F_R$). The memory of the past forces ($F_{old}$) urges the robot to move toward the smaller cloud which was a large cloud once. Although a small cloud of particles means that the chance of finding the target at that area is very low but the target may wander at that area. By removing the small cloud, the robot becomes certain that the target exists somewhere else.

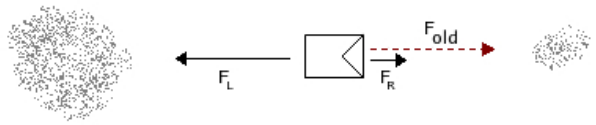



Figure 5.2: One solution to overcome oscillations is to have memory of past forces ($F_{old}$).

The magnitude and direction of the old forces are incorporated in the force calculations according to the following equation:

$$\vec{F^{tot}} = \alpha_1 \vec{F_{new}} + \alpha_2 \vec{F_{old}} \tag{5.4}$$

where $\alpha_1$ and $\alpha_2$ are two constants that are determined according to the structure of the environment before tracking starts and sum to 1. $\vec{F_{new}}$ is the force which is exerted on the robot according to the current location of the particles and the teammate robots and $\vec{F_{old}}$ is specified by the past force memory of the tracker.

# Chapter 6

# Demonstration and Experimental Evaluation

All of the requirements for the proposed method including localization and mapping, probabilistic tracking and path planning, have been discussed. This chapter provides:

1. A simulation demonstration of two robots cooperatively tracking a target in an artificial corridor environment.
2. A simulation demonstration of four robots cooperatively tracking a target in a map based on a real office floor plan at SFU.
3. A simulation experiment to compare the performance of *target tracking* with and without coordination.
4. A simulation experiment to compare the performance of *target finding* with and without coordination with robot teams of different sizes.
5. A validation experiment with two robots in the real world.

Due to limitation of the number of robots and environment maps, most of the experiments are done in simulation using Stage [34].

## 6.1 Player/Stage

As stated before, we use Player/Stage robot server and simulator for our experiments. The authors of Player/Stage describe them thus:

> "Player is a device server that provides a powerful, flexible interface to a variety of sensors and actuators (e.g., robots). Because Player uses a TCP socket-based client/server model, robot control programs can be written in any programming language and can execute on any computer with network connectivity to the robot. Robot controllers are independent client processes communicating with Player via a socket. Player sends a message to each connected controller at 10Hz indicating the corresponding robot's current speed, turn rate, position estimate and sensor readings. Controllers asynchronously send messages back to Player indicating the latest speed and turn rate demands for their robot. In addition, Player supports multiple concurrent client connections to devices, creating new possibilities for distributed and collaborative sensing and control" [8].

Player provides several drivers for robot controllers. Localization and obstacle avoidance are two high level Player drivers that we have used for the experiments of this chapter.

> "Stage is a scalable multiple robot simulator; it simulates a population of mobile robots moving in and sensing a two-dimensional bitmapped environment, controlled through Player. Stage provides virtual Player robots which interact with simulated rather than physical devices. Various sensor models are provided, including sonar, scanning laser range finder, pan-tilt-zoom camera with color blob detection and odometry" [8].

Our Stage models approximate ActivMedia Pioneer-3DX robots with SICK LMS-200 laser range finders for localization and navigation. These are well-known, commonly-used laboratory devices. We also use a Fiducial-finder to detect the objects of interest, modeling a feature detector on a camera or other sensor.

## 6.2 Multiple Robot Target Tracking

In this section, we show simulation examples of target tracking especially the case when the object goes out of field of view of robots' sensors for long periods of time. The robots communicate through TCP and exchange position and orientation information. This information is used for clearing the particles located in other robots' field of view and for computation of the described forces. Usually, a single robot fails to locate the target if the target disappears suddenly. An example case is shown in Figure 6.1. The person has disappeared from the robots view, so the robots don't know the exact position of the target to track it. The only information they have according to the particles' positions is that the target is either on the right or the left side of the corridor. Therefore, the best cooperative action for them is that one robot goes toward a cloud of particles and the other robot goes toward another cloud. This case is difficult for a single robot because of the increase in uncertainty and sometimes results in the failure of tracking.

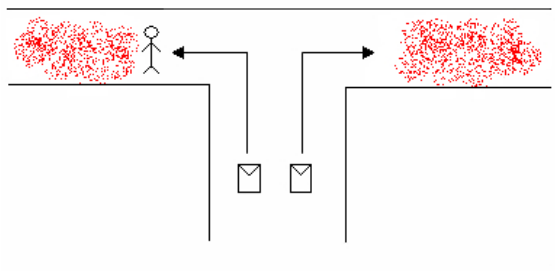


Figure 6.1: Each of the two robots will be attracted to one of the two peaks of the bi-modal distribution.

### 6.2.1 Demonstration 1: Two Robot Example

In this example, shown in Figure 6.2 two searcher robots, $S1$ and $S2$ (start out at the bottom of the world as shown in Figure 6.2($a$) left) try to catch an intruder, $T$ (the robot on the top of the picture shown by blue in color versions). The map is provided to each robot and at the beginning of the process the particles are spread randomly in the traversible free space which means the robots don't have any information about the position of the intruder and the target can be anywhere on the map (Figure 6.2($a$) right). The intruder is controlled manually by the human experimenter who sees the whole world and tries to evade the searchers. The left pictures are snapshots from Stage and the right pictures are the

views of the map and particle cloud maintained by one of the robot controllers at different time steps. The robots start searching by choosing two different directions. Apparently, the chance of finding the intruder would be higher by going from different directions (Figure 6.2($b$)). Then one of the robots sees the intruder but the intruder suddenly goes out of its field of view. Therefore, the particles are gathered in a small region and the searcher robot on the left changes its direction toward the particles so that it reaches the target in the minimum time (Figure 6.2($c$)). The particles start spreading, so the robot on the left changes its direction again to catch the target from its front, while the other robot covers the area behind the intruder (Figure 6.2($d$)). Finally, the searchers catch the intruder and the particle cloud shrinks to a small region which means they have a good estimation of the position of the target (Figure 6.2($e$)).

### 6.2.2 Demonstration 2: Four Robot Example

In this example, we deploy four robots to perform the task of tracking in an environment which is different from the previous map. Instead of narrow corridors, we have several open rooms connected by doorways. Otherwise the experiment is the same as that of previous example. All of the robots communicate and exchange their own and target positions. This case is more difficult than the previous one since there is a high chance that the particles fill in the space which have been visited already and cause the searcher robot to go back and forth between two areas.

Four searcher robots, $S1$, $S2$, $S3$ and $S4$, located on the right side of the first figure, Figure 6.3($a$) (left), try to catch the intruder $T$, which is located behind a wall. Particles are distributed randomly throughout the map except in the regions which are observed by the searchers. As Figure 6.3($b$) shows, each robot chooses a different area for searching. Also, the number of particles behind the wall starts to increase, which means the probability of finding the intruder is now higher in that area. It should be noted that two of the robots, $S1$ and $S3$, choose two different directions to approach the area behind the wall. The robots find the intruder behind the wall and send target location information to other teammates (Figure 6.3($c$) (left)). The particle cloud converge to a single area of the map (Figure 6.3($c$) (right)).

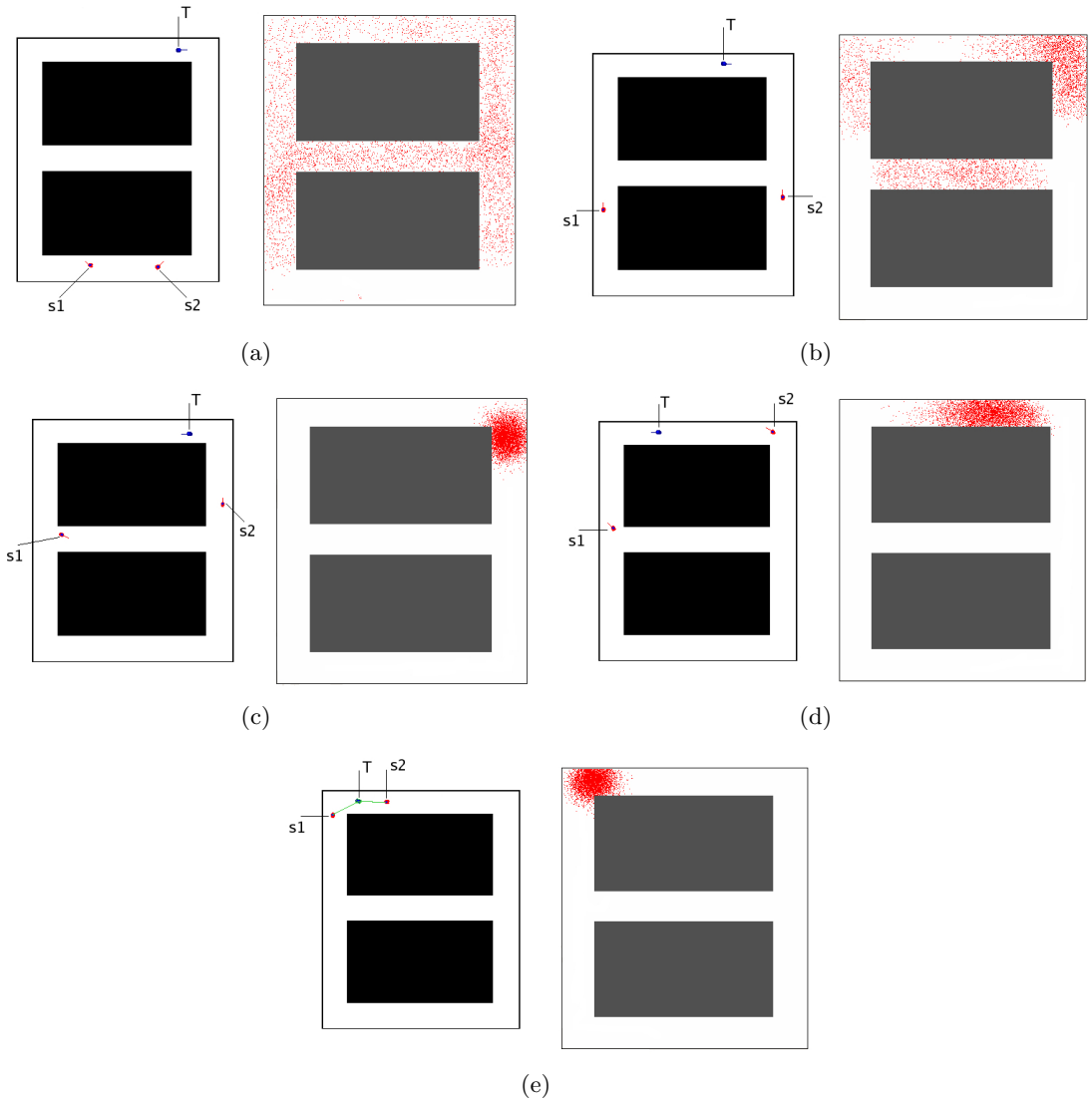
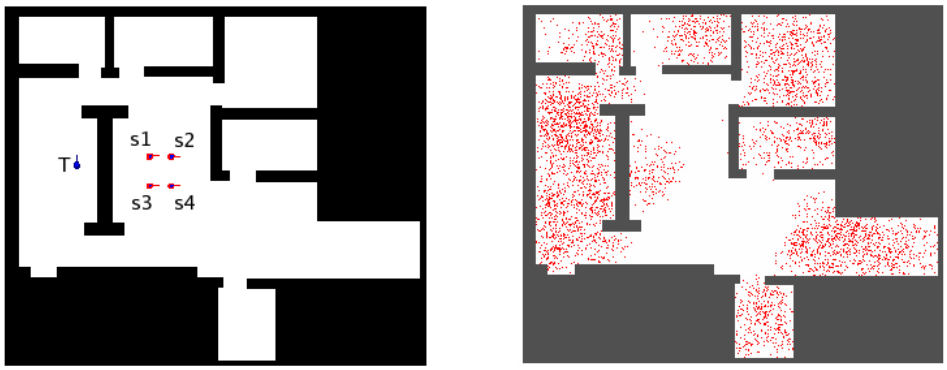
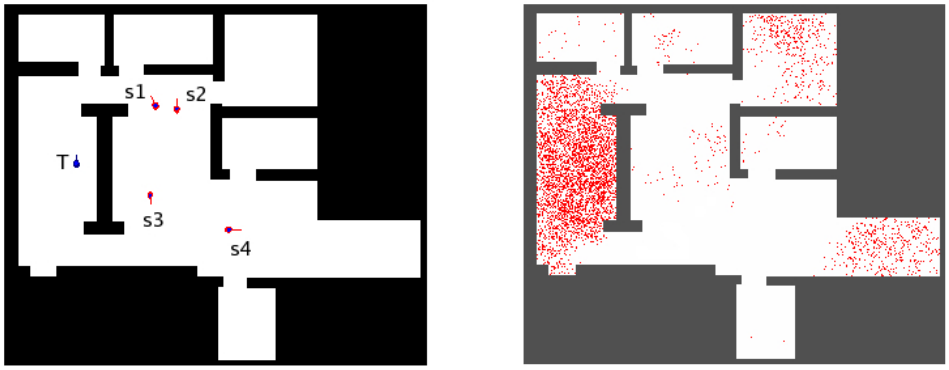


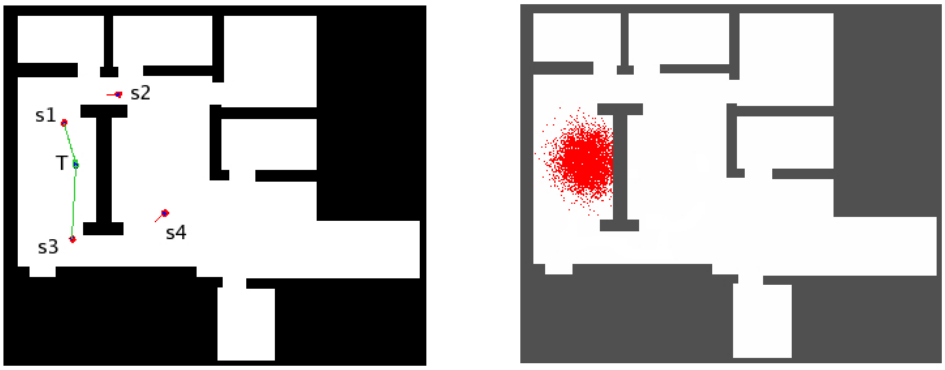

Figure 6.2: Simulation results of two searchers tracking an intruder. S1 and S2 are two searchers that try to find and catch target T.




(a)




(b)




(c)

Figure 6.3: Simulation results of four searchers tracking an intruder. S1, S2, S3 and S4 are four searchers that try to find and catch the stationary target T.

## 6.3 Experiment 1: Evaluation of Tracking

In this section, we evaluate the performance of tracking and uncertainty minimization for a team of robots.

**Goal:** Our goal for performing this experiment is to show:

1. the proposed coordination method will improve the performance of the system which means the amount of time spent for finding the intruder would be decreased and,
2. the time in which the searcher robots have no observation would be less compared to the corresponding times in the experiments without coordination.

**Hypothesis:** We examine if there is any significant difference between the distribution of the mentioned time for different experiments i.e. we are going to:

1. test whether the outcomes of repeated trials of the two methods belong to different distributions.
2. compare the means of the distributions to see if one method is better.

**Method:** In this section, we perform three experiments:

1. Searcher robots try to find the intruder without any coordination strategy. They do not have any information about the teammates and act based on their own decision.
2. The robots communicate but the transfered data is limited to their poses and target observation.
3. The teammate robots share the set of randomly selected particles used in force calculations as well as the pose and observation which means the robots have approximately the same view of the environment. This part requires more communication bandwidth but more information is provided for coordination of the robots. Because of the random nature of particle filters, in some situations, the number of particles may be small in an area from one robot's view while the other robot has large number of particles in that area. Sharing the particles will provide more information to the robots in these situations.

The experiments were performed in an indoor office environment whose map is represented in Figure 6.3 to show the performance of the coordination methods compared to the

case when the robots do not cooperate. There are two searcher robots in these experiments that try to catch the intruder by cooperative uncertainty minimization. The intruder moves randomly from a random start position. It chooses a random location on the map as goal points and heads toward that location while avoiding obstacles using VFH [2]. After a fixed amount of time $t$, a new goal location is selected. In these experiments, $t = 5$ *sec*.

The start position of the searchers is the same in all of the experiments. For these experiments, the range of the robot sensors is set to be 8 meters with an angle of view of 120 degrees while the environment dimension is $24m \times 20.5m$.

**Results:** The results shown in Figure 6.4 were gathered from two robots in 10 trials of five minutes of tracking (20 samples for each experiment in total). The light gray area shows the average percentage of time a searcher spends before first locating the intruder. That means, how successful were the teammate robots in decreasing the uncertainty of initially uniformly distributed particles. The dark gray area is the average percentage of time that the searchers had no observation after they see the intruder for the first time.

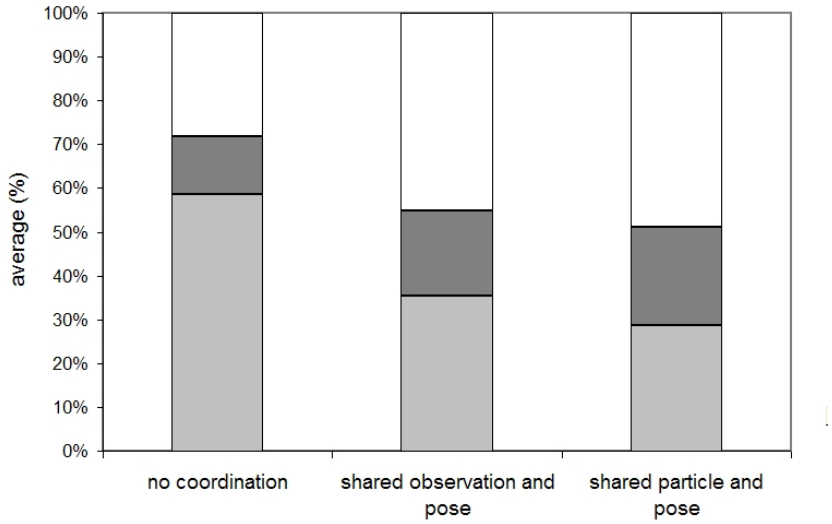



Figure 6.4: No coordination, Shared observation and pose and Shared particles (from left to right) are three cases shown in this diagram. The light gray area shows the average percentage of time an agent has spent before visiting the object for the first time. The dark gray area is the average percentage of time that the robots had no observation after they see the intruder for the first time [25]. Printed by permission of IEEE ©.

Table 6.1 and 6.2 also show the mean and the standard deviation for the time that the searchers had no observation before visiting the intruder for the first time and the total

time without any observation, respectively. The fourth column also shows the p-value for the T-test for comparing each experiment with the non-coordinated experiment. T-test is a statistical test that shows if two normally distributed data sets belong to distributions with equal means.

| Trial Type | Mean (sec) | $\sigma$ | T-test $p$-value |
|---|---|---|---|
| No coordination | 175.7 | 99.5 | — |
| Shared Observation and Pose | 106.5 | 85.3 | 0.0076 |
| Shared Observation, Pose and Particles | 85.7 | 54.5 | 0.0009 |

Table 6.1: Mean, Standard Deviation and the result of T-test is shown for the time that the searchers had no observation before visiting the intruder for the first time. Since the $p$-values are less than 0.05 we can conclude that the coordinated methods performed better.

| Trial Type | Mean (sec) | $\sigma$ | T-test $p$-value |
|---|---|---|---|
| No coordination | 215.8 | 70.3 | — |
| Shared Observation and Pose | 164.6 | 63.7 | 0.0208 |
| Shared Observation, Pose and Particles | 153.4 | 53.3 | 0.0046 |

Table 6.2: Mean, Standard Deviation and the result of T-test is shown for the total time that the searchers had no observation. Since the $p$-values are less than 0.05 we can conclude that the coordinated methods performed better.

**Conclusion:** The diagram shows the total time that the robots have no observation (sum of the values of light and dark gray areas) in the shared-particle and non-shared particle case is less than that of no-coordination case, indicating that the performance is improved on average by cooperation. In addition, since the $p$-values of the T-test is less than 0.05, we can conclude that the data do not belong to distributions with equal means and there is a significant difference between the performance of non-coordinated and coordinated cases where coordinated cases have performed better. The large standard deviations are an expected issue in this experiment because the starting position of the intruder was different in the trials and the amount of time required for finding the target can vary greatly. Therefore,

the average time used as a measure for general comparisons.

## 6.4 Experiment 2: Evaluation of Communication Effect

In this experiment, we evaluate the performance of uncertainty minimization for different populations of robots in coordinated and non-coordinated cases. The goal of the robots is to minimize entropy of particles and find the intruder.

**Goal:** Our goal is to show that communication among the robots will decrease the time for finding a target given an initially uniformly distributed set of particles. Also, increasing the number of the robots will cause a significant difference between the performance of robot teams with different population sizes.

**Hypothesis:** We examine if there is any significant difference between the performance of uncertainty minimization in coordinated and non-coordinated cases. By coordinated we mean the robots exchange pose information (no particle information) and in the non-coordinated case each robot chooses an action independent of the decisions of their teammates. We also check if increasing the number robots has any significant effect on the performance.

**Method:** We performed 50 trials of this experiment and tested the effect of communication for two, three and four searcher robots. The one-robot case was also performed for comparison purposes. Initial positions of searchers and the intruder are random and the intruder does not have any intelligence and its movement is completely random as well. The intruder selects a cell on the map occupancy grid (occupied or unoccupied) at random, then it chooses another cell after a fixed period of time. The obstacle avoidance method (VFH) will cause the robot to move around the obstacle if an occupied cell is chosen. The environment which is adopted for this experiment is part of simplified map of SFU TASC building which is shown in Figure 6.5. It consists of four loops which makes the search more difficult.

Figure 6.5: Simplified map of part of SFU TASC building. The dimension is $70m \times 19m$.

The map dimension is $70m \times 19m$ and the size of each cell in the occupancy grid is $15cm \times 15cm$. Each robot carries a laser range scanner with field of view of up to 8 meters at 180 degrees in its front. Fiducial-finders are used to detect the target. The searcher robots move at $0.2ms^{-1}$ and the intruder moves at $0.3ms^{-1}$ at the normal condition (it may stop for a while if it reaches a target location before selecting a new location). The pseudo random numbers generator seed is the same for corresponding trials. Therefore, for example, initial positions and traversed path of the intruder were the same in the first trial of all of the experiments. It should be noted that the intruder client is subject to stochasticity in the TCP connection which might cause VFH to choose different paths in the corresponding trials.

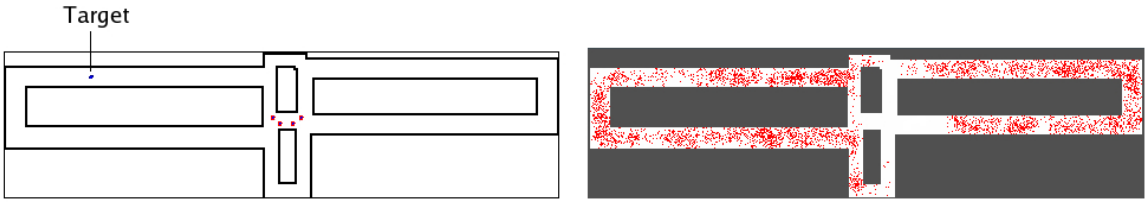
The experiments start with randomly distributed particles on the map and the searchers continue until the intruder is found or a time limit reached. Four hundred seconds was the time limit considered for these experiments. Initially randomly distributed particles is the worst case for the tracker which means the searchers initially have no idea where the intruder is located.

**Results:** The results show that in the coordinated robots cases, the robots try to keep their distances as far as possible from each other considering the distribution of the particles and in the non-coordinated robots cases, the searchers go to the middle of the map which is the most possible area for capturing the intruder (Figure 6.6).

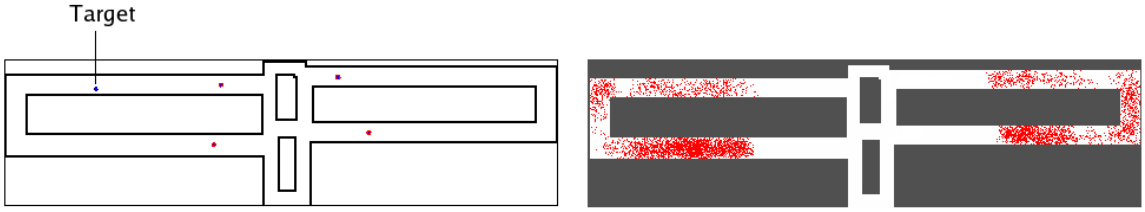
The time histograms of the experiments are shown in figures 6.7, 6.8, 6.9 and 6.10. Statistical tests were performed to check if there is any significant difference between coordinated and non-coordinated cases and also to examine if increasing number of robots has any effect on the performance.

We chose Wilcoxon Signed-Rank test to analyze the data. This test is equivalent to T-test for non-parametric data. The details of the test are explained in Appendix B. In summary, this test ranks the absolute differences between paired data, then it examines if the median of the differences is greater or less than zero according to the signed sum of the ranks. If the $p$-value is less than 0.05, we can conclude that there is a significant difference between two distributions.

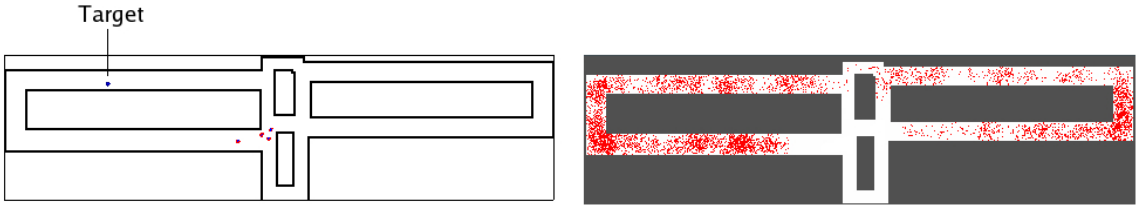
**Conclusion:** As table 6.3 shows, there is significant statistical difference between non-coordinated and coordinated cases for four and three robots but the coordination strategy does not improve the performance of tracking for two robots in that environment. One of the reasons for not being significantly different for two robots experiment is that the




(a) $t = 0$




(b) $t = 50\,sec$

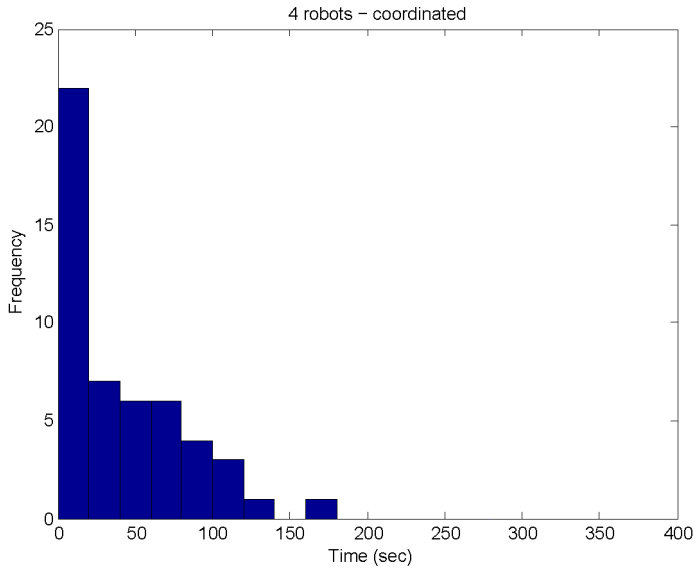



(c) $t = 50\,sec$

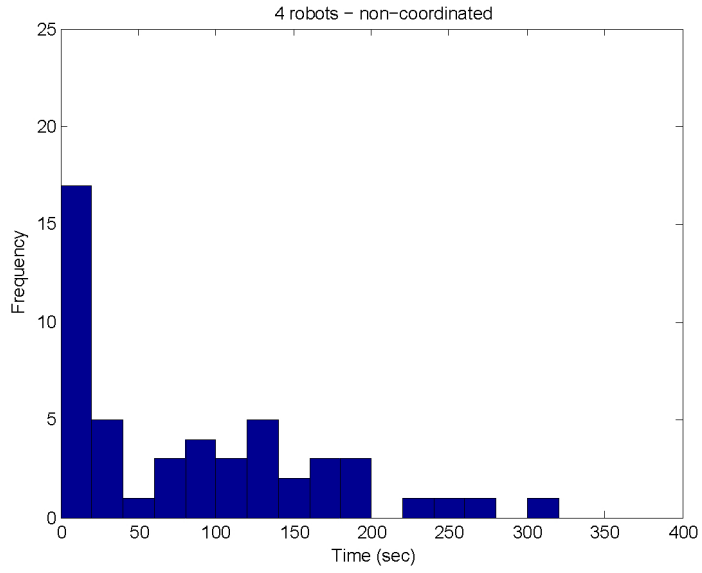
Figure 6.6: (a) This figure shows a start configuration of robots in the experiment that four robots try to locate an intruder. Left pictures are Stage snapshots and right pictures are view of one of the robots. (b) Robots configuration after a certain amount of time for coordinated robots. (c) Robots configuration after a certain amount of time for non-coordinated robots.

| Number of Robots | $z$ ratio | $p$ value | Sum of Neg. Ranks | Sum of Pos. Ranks |
|---|---|---|---|---|
| Four | -4.518 | 0.000 | 59.00 | 682.00 |
| Three | -2.540 | 0.011 | 234.50 | 626.50 |
| Two | -0.942 | 0.346 | 475.00 | 653.00 |

Table 6.3: Wilcoxon test results to check if there is any difference between coordinated and non-coordinated cases for different population sizes.

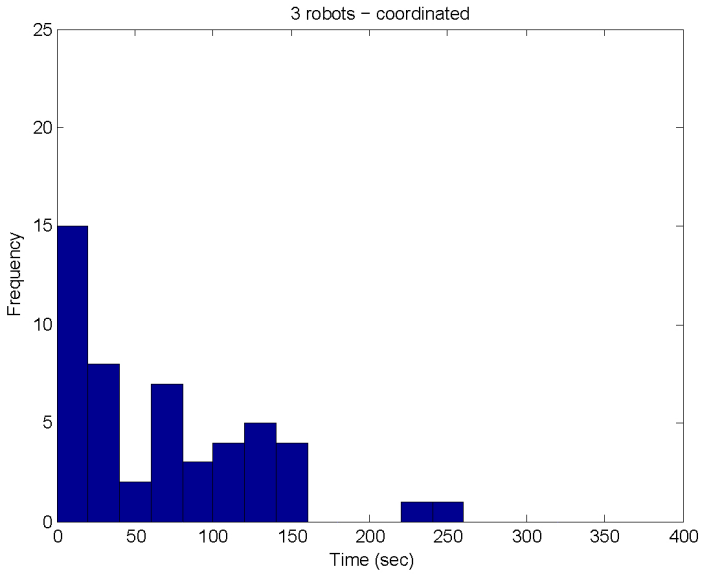



(a)

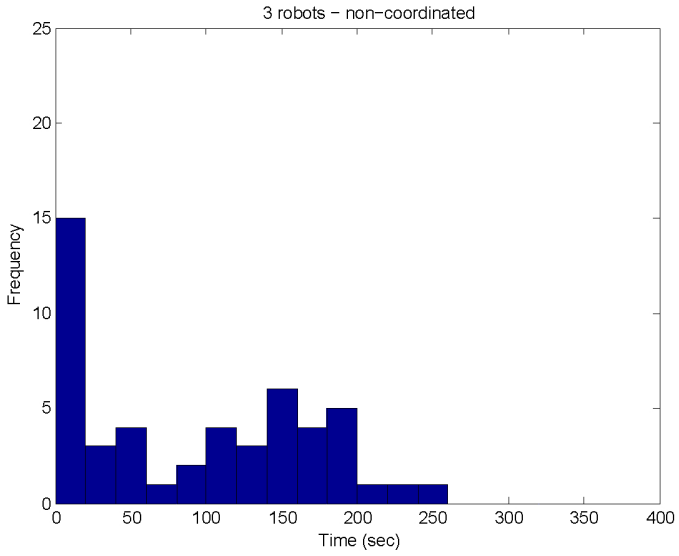



(b)

Figure 6.7: Histograms for four-robot experiments in which the robots were supposed to locate an intruder in SFU TASC map with and without a coordination strategy.

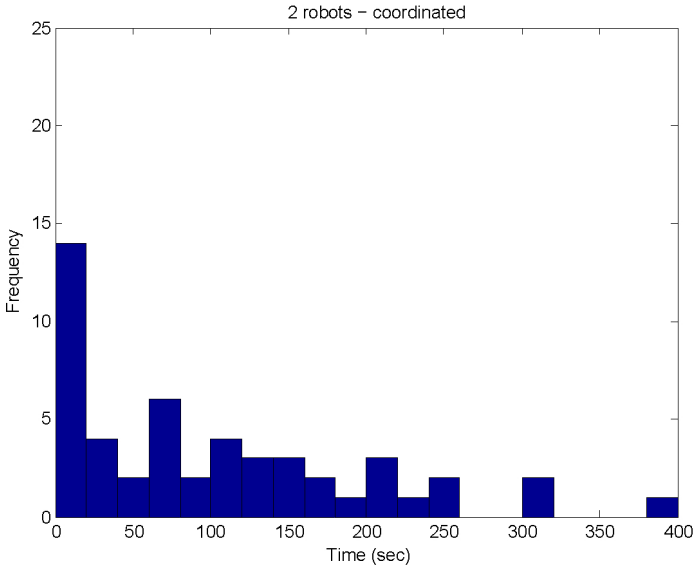



(a)

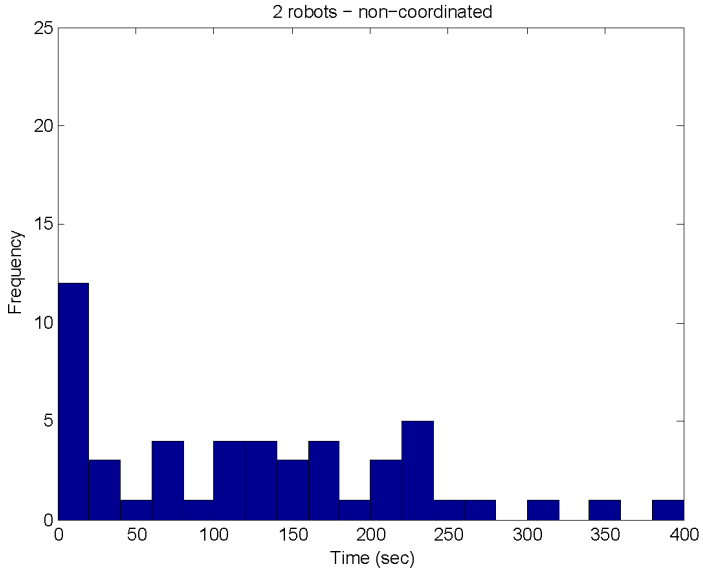



(b)

Figure 6.8: Histograms for three-robot experiments in which the robots were supposed to locate an intruder in SFU TASC map with and without a coordination strategy.




(a)




(b)

Figure 6.9: Histograms for two-robot experiments in which the robots were supposed to locate an intruder in SFU TASC map with and without a coordination strategy.

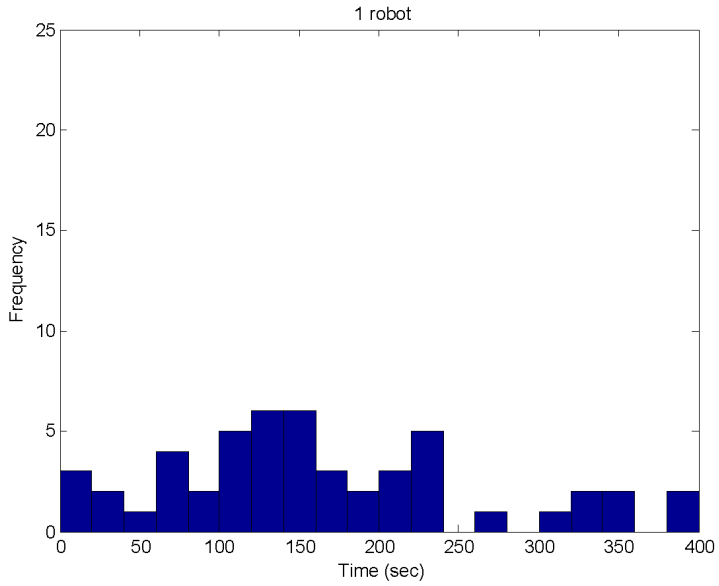



Figure 6.10: Histograms for one-robot experiment in which the robot was supposed to locate an intruder in SFU TASC map.

| N of Robots | $z$ ratio | $p$ value | Sum of Neg. Ranks | Sum of Pos. Ranks |
|---|---|---|---|---|
| 4 vs. 3 | -4.283 | 0.000 | 170.50 | 1005.50 |
| 4 vs. 2 | -4.949 | 0.000 | 105.50 | 1070.50 |
| 4 vs. 1 | -6.144 | 0.000 | 1.00 | 1274.00 |
| 3 vs. 2 | -2.241 | 0.025 | 369.50 | 806.50 |
| 3 vs. 1 | -4.614 | 0.000 | 159.50 | 1115.50 |
| 2 vs. 1 | -3.171 | 0.002 | 309.00 | 966.00 |

Table 6.4: Wilcoxon test results to check if there is any advantage to increase the number of robots.

| Number of Robots | $z$ ratio | $p$ value | Sum of Neg. Ranks | Sum of Pos. Ranks |
|---|---|---|---|---|
| Four | -3.750 | 0.000 | 162.50 | 783.50 |
| Three | -4.905 | 0.000 | 83.50 | 952.50 |
| Two | -4.493 | 0.000 | 110.00 | 880.00 |
| One | -3.142 | 0.002 | 312.00 | 963.00 |

Table 6.5: Comparison of each experiment with its corresponding best case.

environment is too large for two robots to effectively clear the particles and the particles fill in the previously cleared areas very quickly and a higher level planning is required.

We compared the results of the experiments for different number of robots for coordination case. For each pair of population sizes, we aim to establish whether the performance data is drawn from distributions with significant difference. Table 6.4 shows there is a significant statistical difference among the results of the experiments for different number of robots which means increasing the number of robots has improved the performance of cooperative uncertainty minimization.

The results show that in some trials, a lesser number of robots have performed better compared to trials with increased number of robots but these better performances are usually due to the noise in the robot controllers. The differences between times of finding the target are very small (Wilcoxon test also assigns a low rank to these small differences) and in general having more robots will decrease the time spent on finding the intruder.

Finally, we compared the result of experiments for different number of robots that had coordination with the corresponding best cases. By the best case, we mean each robot knows the perfect position of the intruder. The comparison result is shown in Table 6.5. Wilcoxon Signed-Rank test shows the performances measured in the best cases are significantly different and better.

## 6.5 Experiment 3: Real Robot Experiment

This section describes implementation of the proposed method on real robots.

**Hypothesis:** We examine if the proposed method will work in practice using real robots. The main challenge which differentiates this experiment from the previous ones, is dealing with noise in robot sensors and actuators that is not present in simulation.

**Method:** After collecting map data for robots, we run experiments just by copying the code used for simulated robots to the real robots to examine the proposed approach works well in the real world. We used two Pioneer-3DX robots (shown in Figure 6.11) with the following specifications:

- Two-wheel differential drive,
- Equipped with SICK LMS-200 laser scanner that can measure distances up to 8 meters very accurately. Range resolution $\approx 2\,mm$ and angular resolution $\approx 0.5\,degrees$,

- Equipped with a sonar ring consisting 16 sonar sensors around the robot (not used in our experiment due to their high level of noise),
- Equipped with a 802.11$b$ tele-communication module used for communication among the robots,
- Maximum speed: 1.6 $ms^{-1}$.

Figure 6.11: Pioneer-3DX robots. Printed by permission of Richard Vaughan.

The task which is defined for robots in this experiment is to cooperatively minimize uncertainty to capture the intruder in the corridors or to make sure there is no intruder on the portion of the map provided to the robot. The corridor map provided to the robots is shown in Figure 6.12. The particles are distributed uniformly on the entire map at the beginning which means the intruder can be anywhere on the map. If the robots clear all of the particles, they can be certain that there is no intruder inside that part of the building (for our experiment no intruder is present).

**Result:** The snapshots of the real robot experiment are shown in Figure 6.15. The snapshots show one of the robots at five different points on the traversed path. Figure 6.15($a$) shows the start of the experiment with two robots. The robots exhibit desirable behaviour: the searcher robots go toward the particles from two different directions to find the target faster. Due to small number of trials and limited number of robots no histograms and comparisons are given for these tests.

**Discussion:** One of the major advantages of using Player/Stage is that the client code can be transfered to a real robot without any change. The results show that transferring a

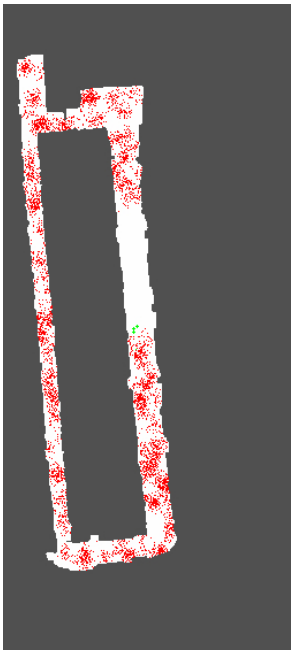


Figure 6.12: The map provided to the robots for the experiment is shown. The white area means that the robots have cleared that part at the start of tracking.

simulated client code to a real robot worked well and the robots showed the similar behaviour in practice. There are some issues about the real robot experiment that should be discussed. One of the major problems in real robot experiments is the range of communication. As the robots go out of communication range of teammates, they can not cooperate with others which really degrades the performance of the system. We measured the wireless signal strength on the previously shown map and combined that with the AMCL (Adaptive Monte Carlo Localization) data to check the communication range. The strength diagram is shown in Figure 6.14, the strength is measured from the starting position of the robots in the mentioned experiment.

As shown in the diagram, the signal is weak in the far distances which might cause problems for robots to communicate with teammate robots. One of the solutions to this problem for large buildings is to use a repeater to boost the signal or to use more powerful antennas.

The other problem is due to the materials that do not reflect laser beams. For instance, laser beam passed through the glass surrounded SFU TASC corridors and since the localization is based on laser scanner data, the error of localization increased at some points.

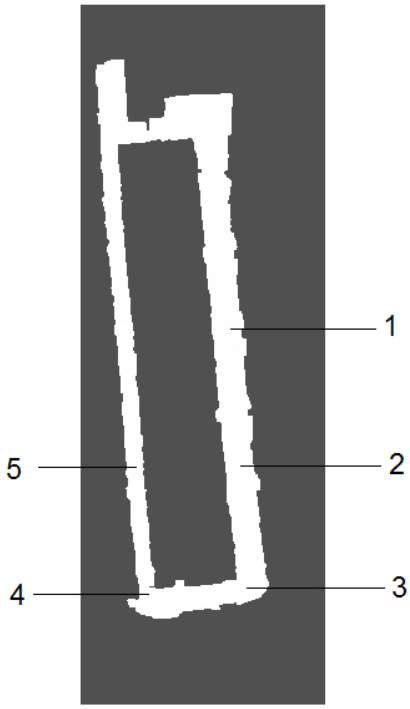



Figure 6.13: Five different points on the traversed path of one of the robots in the real robot experiment are marked. Corresponding snapshots are shown in Figure 6.15.

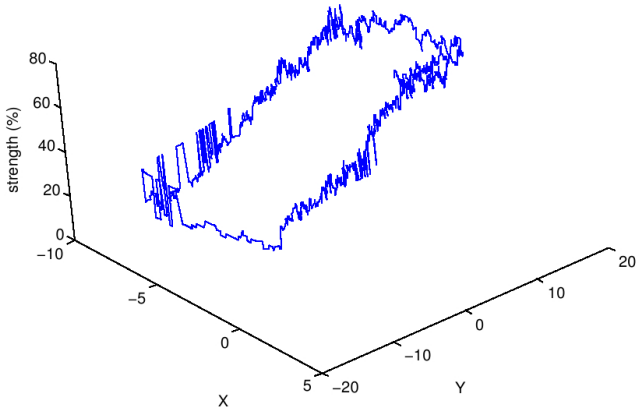



Figure 6.14: Wireless signal strength diagram obtained in SFU TASC building. $X$ and $Y$ are provided in meters.

(a) Snapshot taken at point 1

(b) Snapshot taken at point 2

(c) Snapshot taken at point 3

(d) Snapshot taken at point 4

(e) Snapshot taken at point 5

Figure 6.15: Snapshots of real robot experiment with two robots in SFU TASC building. The corresponding points on the map are shown in Figure 6.13.

# Chapter 7

# Conclusion and Future Work

## 7.1 Summary

A method for a team of mobile robots to cooperatively track a moving target has been described in this thesis. The method and part of experimental results have been published as [25]. My contribution is to address the main limitation of previous approaches in that it actively minimizes the uncertainty caused when the target is occluded for long periods. A particle filtering method represents the multi-modal uncertainty in the estimated pose of the target. Then a potential field is generated using the location of particles directly as input - no clustering of particles is performed. The potential field guides the robots to visit as many particles as they can to reduce the uncertainty in the environment and to prevent the uncertainty area from further growing. The integrated particle filter and potential field method performs as a path planner at the same time.

The algorithm is extended to multiple robots by allocating subsets of particles to each robot. A simple nearest-robot filter is used to achieve this. Also, probabilistic robot mapping and localization as the prerequisites for the proposed method were explained in the first few chapters of this thesis.

An experimental section then showed examples of cooperative tracking using multiple robots in different environments and compared the performance of a pair of robots tracking a target without coordination, and with two alternative coordination methods. Then comparisons for different population sizes and the effect of communication among robots showed that in most of the cases increasing the number of robots and having communication will improve performance of tracking. Additionally, the real world results for two robots showed

that the new tracking method can be applied to real robots as well as simulated robots.

The method described tracking of a single target but by using methods from the literature cited in the background section, the method can be easily extended to multi-target problems with the caveat of exponential complexity .

The novel tracking technique has some shortcomings and there are some suggestions for future extension of this work that should be discussed.

## 7.2 Optimality of the Solution

The first deficiency of this method is that it is based on a heuristic search and it is not the optimal solution to this problem and certain cases might happen that the searchers are not able to locate the intruder. However, to the best of my knowledge this approach is the most efficient approach to this issue so far. As mentioned throughout the thesis, there are some optimization frameworks such as POMDPs [29] to get an optimal solution for this problem but they have not shown any success in solving this problem since they are computationally intractable.

## 7.3 Local Optima Issues

The second criticism of this method is that it is subject to rapidly changing local optima. An example case was shown in Figure 4.7 where the problem was that the robot was in the halfway distance between a large and a small cloud of particles. The robot was attracted toward the larger cloud but due to clearing of the particles, the large cloud became small and the small cloud became large and the robot changed its direction to the current large cloud. This problem can cause an oscillatory behaviour in the navigation of the robot. A trick was proposed for overcoming this problem by remembering the past direction of attracting forces. However, that solution is not a complete solution to this problem.

## 7.4 Full Communication Problem

Another drawback of this method is that its performance relies on full communication among robots which might be impossible in some situations. One problem was discussed in the real robot experiment section where the wireless signal strength was too weak at far

distances. Although the robots can decide based on their own information, the performance of tracking and uncertainty minimization will degrade. One possible extension to this work is that each robot localizes teammate robots in its coordinate frame instead of relying on full communication. This extension is possible according to the current literature on localization. In this case, the performance of the system will not change significantly if one robot looses communication with other robots.

## 7.5 Efficiency of the Flood-Fill Method

As described in chapter four, the magnitude and direction of particle forces are computed using a flood-fill method. The time complexity of this method for calculation of map distances is proportional to the size of the map and occupancy grid cells. Therefore, due to the computational cost of this method, it is not feasible to have a measure between two individuals or sets of particles which might be helpful to improve tracking. A good extension is to replace this method with a more efficient one. For instance, we can use a pre-computed lookup table for every possible position of a robot on a map instead of online calculations.

## 7.6 Environment and Initial Positions

One of the main factors that influences most robotic systems is the environment in which the robot works. The performance of our system is also subject to the environmental constraints. As stated, having coordination and increasing the number of robots will improve the performance of the system significantly in the mentioned experiments but in some environments the improvement may not be significant. For example, we did not see a 'significant' improvement for large populations in experiments performed on the map shown by Figure 6.3. The reasons are the small size of the environment and its open area structure. Since the chance of visiting an intruder in an open area is higher, the searchers may find the intruder quickly even without communication.

Another factor on which our proposed method is dependent is the start position of searchers. In certain start positions, the performance of the system would be much better. For example, Figure 6.6($a$) shows a good start position in the shown map.

## 7.7 Target Motion Model

For the experiments of this thesis we used an empirical motion model of the target. Another future extension of this work is that the parameters of the motion model are estimated online. For instance, vision-based activity recognition methods can be used to estimate the current activity of humans (running, walking, etc.) [6] and the motion model would be adjusted according to these data.

## 7.8 Simultaneous Localization and Mapping

We separated mapping and localization steps due to the available tools and for reducing the computational burden of the system. It would be more interesting if the robots perform mapping and localization simultaneously instead of having *a priori* map but it should be done in an efficient way. Like the method which is mentioned in [22], we can condition the target particles upon localization particles and reduce the size of particle sets.

# Appendix A

# Vector Field Histogram

VFH (Vector Field Histogram) [2] is a method of fast obstacle avoidance for mobile robots equipped with range sensors. This section explains this method and its shortcomings very briefly.

In this method, we consider a local square window around the robot where the window contains a grid called histogram grid. Each cell in the grid will be assigned a certainty value that represents the confidence in existence of an obstacle at that location. The reason for using confidence level is that, this method first introduced for ultra-sonic sensors. To account for sensors' shortcomings such as inaccuracy and crosstalk, they consider a certainty value. So each time the sensor detects an obstacle in a location the value of the corresponding cell will be increased. Then, a one dimensional *polar histogram* is constructed around the robot's momentary location. This histogram comprises $n$ angular sectors of width $\alpha$. The contents of each cell in the window is mapped into the corresponding sector of the polar histogram. That means a sector that contains many cells with high certainty value, will be assigned a higher value. Figure $A.1$ shows the polar histogram for the robot in Figure $A.2$.

A threshold on the polar histogram determines the candidate directions for subsequent travel. Candidate directions are shown as dark gray in Figure $A.2$, while unsafe directions (i.e., those with polar obstacle densities above the threshold) are shown in light gray. Usually there are several candidate directions and the VFH algorithm selects the one that most closely matches the direction to the target. The main deficiency of this method is that it is subject to local optima causing oscillatory behavior. In our case, the planner and navigation controller on top of VFH have solved the shortcomings of the underlying obstacle avoidance device.

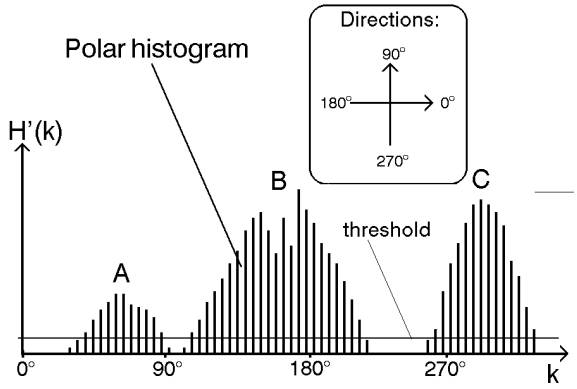



Figure A.1: The polar histogram corresponding to the momentary position of the robot at point. The value of the bars are functions of the distance from obstacles [2]. Printed by permission of Johann Borenstein ©.

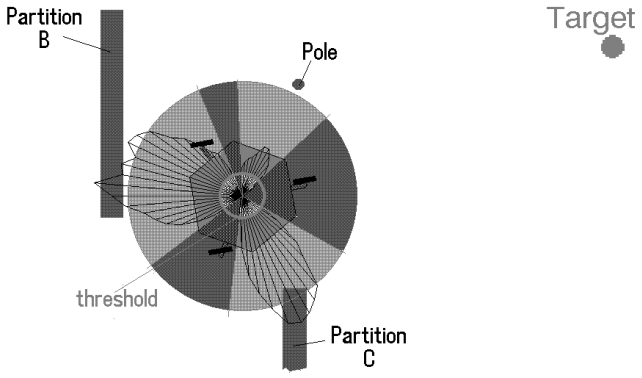



Figure A.2: Dark gray regions are candidate directions for navigation [2]. Printed by permission of Johann Borenstein ©.

# Appendix B

# The Wilcoxon Signed-Rank Test

> "The Wilcoxon matched-pairs signed ranks test is an equivalent of t-test designed for non-parametric data and fulfills the need for the case of two related samples when the measurement scale allows us to determine not only whether the members of a pair of observations differ, but also the relative magnitude of any difference. In other words, the Wilcoxon matched-pairs signed-rank test is appropriate when we can determine the amount of any difference between pairs of observations $X_i$ and $Y_i$, as well as the direction of the difference. When we can determine the magnitudes of differences, we can rank them. It is through the rankings of the differences that the Wilcoxon test utilizes the additional information (compared to regular sign test)" [3].

An example case is shown by table $B.1$ when the number of pairs are equal to 16. The Wilcoxon test begins by transforming each instance of $X_i - Y_i$ into its absolute value, which is accomplished simply by removing all the positive and negative signs. Thus the entries in column 4 of the table below become those of column 5. Two types of tied observations may arise when using the Wilcoxon signed-rank test: 1) Observations in the sample may be exactly equal (i.e. 0 in the case of paired differences). Such pairs are eliminated from consideration, since they provide no useful information and the number of pairs is adjusted accordingly. 2) Two or more differences may be equal. If so, the average of the ranks will be replaced. Absolute differences are then ranked from lowest to highest.

The result of this step is shown in column 6. The entries in column 7 will then give us the clue to why the Wilcoxon procedure is known as the signed-rank test. The same entries

as in column 6 are seen, except now they have been attached the positive or negative sign that was removed from the $X_i - Y_i$ difference in the transition from column 4 to column 5.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| $i$ | $X_i$ | $Y_i$ | original $X_i - Y_i$ | absolute $X_i - Y_i$ | rank of absolute $X_i - Y_i$ | Signed Rank |
| 1 | 78 | 78 | 0 | 0 | — | — |
| 2 | 24 | 24 | 0 | 0 | — | — |
| 3 | 64 | 62 | +2 | 2 | 1 | 1 |
| 4 | 45 | 48 | -3 | 3 | 2 | -2 |
| 5 | 64 | 68 | -4 | 4 | 3.5 | -3.5 |
| 6 | 52 | 56 | -4 | 4 | 3.5 | -3.5 |
| 7 | 30 | 25 | +5 | 5 | 5 | 5 |
| 8 | 50 | 44 | +6 | 6 | 6 | 6 |
| 9 | 64 | 56 | +8 | 8 | 7 | 7 |
| 10 | 50 | 40 | +10 | 10 | 8.5 | 8.5 |
| 11 | 78 | 68 | +10 | 10 | 8.5 | 8.5 |
| 12 | 22 | 36 | -14 | 14 | 10 | -10 |
| 13 | 84 | 68 | +16 | 16 | 11 | 11 |
| 14 | 40 | 20 | +20 | 20 | 12 | 12 |
| 15 | 90 | 58 | +32 | 32 | 13 | 13 |
| 16 | 72 | 32 | +40 | 40 | 14 | 14 |

Table B.1: This table shows the ranks for 16 pairs of data. $N = 14$, $W^+$(sum of positive ranks)= 86, $W^-$(sum of negative ranks)= 19 and $W = W^+ - W^- = 67$ [19]. Used by permission of Richard Lowry ©.

Critical values of the test statistics are available as tables in [3]. The value which is usually considered for acceptance or rejection of hypotheses is $|z| = 1.96$. These tables are useful for small number of paired data. When the number of samples is greater than 25, test statistics are obtained by:

$$z = \frac{W - [n(n+1)]/4}{\sqrt{n(n+1)(2n+1)/24}} \tag{B.1}$$

where $W$ is either $W^+$ or $W^-$ (depending on the problem) and $n$ is the number of paired samples. The critical values of $z$-ratio and their corresponding probabilities ($p$ value) can be obtained from table $B.2$. All of the Wilcoxon Signed-rank tests in this thesis have been performed by using SPSS software which provides useful tools for various kinds of

| $p$ (one-tailed) | 0.05 | 0.025 | 0.01 | 0.005 | 0.0005 |
|---|---|---|---|---|---|
| $p$ (two-tailed) | — | 0.05 | 0.02 | 0.01 | 0.001 |
| $z$ | 1.645 | **1.960** | 2.326 | 2.576 | 3.291 |

Table B.2: Critical values of $\pm z$ for one-sided and two-sided tests [19]. Used by permission of Richard Lowry ©.

statistical tests. It should be noted that non-coordinated experiment data and the data from experiments with less number of robots are assumed to be $X_i$ in this explanation.

# Bibliography


[1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[2] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, 1991.

[3] W. W. Daniel. *Applied Nonparametric Statistics*. Houghton Mifflin Company, 1978.

[4] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10:197–208, 2000.

[5] G. Dudek, M. Jenkin, and E. Milios. A taxonomy of multirobot systems. In T. Balch and L. E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*, pages 3–22. A K Peters, 2002.

[6] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proceedings of the 9th International Conference on Computer Vision*, volume 2, pages 726–733, 2003.

[7] D. Fox. Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research (IJRR)*, 22(12):985–1004, 2003.

[8] B. Gerkey, R. Vaughan, and A. Howard. *The Player/Stage Project*. February 2006. Available online at: `http://playerstage.sourceforge.net`.

[9] B. P. Gerkey, S. Thrun, and G. Gordon. Visibility-based pursuit-evasion with limited field of view. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 2004)*, pages 20–27, San Jose, California, 2004.

[10] B. P. Gerkey, R. T. Vaughan, K. Stoy, A. Howard, G. S. Sukhatme, and M. J. Mataric. Most valuable player: A robot device server for distributed control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1226–1231, 2001.

[11] A. Howard. *Simple Mapping Utilities (pmap)*. February 2006. Available online at: `http://www-robotics.usc.edu/~ahoward/pmap/`.

[12] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal on Computer Vision*, 29(1):5–28, 1998.

[13] B. Jung and G. S. Sukhatme. A generalized region-based approach for multi-target tracking in outdoor environments. In *IEEE International Conference on Robotics and Automation*, pages 2189–2195, New Orleans, LA, 2004.

[14] Z. Khan, T. Balch, and F. Dellaert. A Rao-Blackwellized particle filter for eigentracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 980–986, Washington, DC, 2004.

[15] Z. Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(11):1805–1918, 2005.

[16] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[17] S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *IEEE International Conference on Robotics and Automation*, pages 737–742, 1997.

[18] B. Limketkai, L. Liao, and D. Fox. Relational object maps for mobile robots. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1471–1476, 2005.

[19] R. Lowry. *Concepts and Applications of Inferential Statistics*. February 2006. Available online at: `http://faculty.vassar.edu/lowry/webtext.html`.

[20] M. Matsuoka, S. Singh, A. Chen, A. Coates, A. Y. Ng, and S. Thrun. Autonomous helicopter tracking and localization using a self-calibrating camera array. In *Proceedings of the Fifth International Conference on Field Service Robotics*, 2005.

[21] M. Mazo, A. Speranzon, K. H. Johansson, and X. Hu. Multi-robot tracking of a moving object using directional sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1103–1108, New Orlean, LA, 2004.

[22] M. Montemerlo., W. Whittaker, and S. Thrun. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 695–701, Washington, DC, 2002.

[23] H. P. Moravec and A. E. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 116–121, 1985.

[24] R. Mottaghi and S. Payandeh. An overview of a probabilistic tracker for multiple cooperative tracking agents. In *the 12th International Conference on Advanced Robotics*, pages 888–894, Seattle, WA, 2005.

[25] R. Mottaghi and R. Vaughan. An integrated particle filter & potential field method for cooperative robot target tracking. In *IEEE International Conference on Robotics and Automation*, Orlando, FL, 2006. To appear.

[26] N. Papanikolopoulos, P. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE Transactions on Robotics and Automation*, 9(1):14–35, February 1993.

[27] L. E. Parker. Current state of the art in distributed autonomous mobile robotics. In L. E. Parker, G. Bekey, and J. Barhen, editors, *Distributed Autonomous Robotic Systems 4*, pages 3–12. Springer-Verlag, 2000.

[28] M. Rosencrantz, G. Gordon, and S. Thrun. Locating moving entities in dynamic indoor environments with teams of mobile robots. In *Proceedings of Autonomous Agents and Multi-Agent Systems*, pages 233–240, Melbourne, Australia, 2003.

[29] N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

[30] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. *International Journal of Robotics Research (IJRR)*, 22(2), 2003.

[31] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, pages 1–35. Morgan Kaufmann, 2002.

[32] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[33] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[34] R. Vaughan, B. Gerkey, and A. Howard. On device abstractions for portable, reusable robot code. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2421–2427, Las Vegas, USA., 2003.

[35] M. Zuluaga and R. Vaughan. Reducing spatial interference in robot teams by local-investment aggression. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2798–2805, Edmonton, Alberta, 2005.