UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Active Cooperative Perception in Networked Robot Systems

ABDOLKARIM PAHLIANI

SUPERVISOR : DOCTOR PEDRO MANUEL URBANO DE ALMEIDA LIMA

Jury

THE PRESIDENT OF THE IST'S SCIENTIFIC BOARD

DOCTOR PEDRO MANUEL URBANO DE ALMEIDA LIMA

DOCTOR PEDRO MANUEL BRITO DA SILVA GIRAO

DOCTOR JORGE MANUEL MIRANDA DIAS

DOCTOR ALESSANDRO FARINELLI

DOCTOR ALEXANDRE JOSE MALHEIRO BERNARDINO

DOCTOR MATTHIJS THEODOR JAN SPAAN

Thesis specifically prepared to obtain the PhD Degree in

Electrical and Computer Engineering

PASS

MARCH 2012

Resumo

Esta tese aborda problemas de percepção cooperativa e percepção cooperativa activa em redes de sistemas robóticos. O problema de percepção cooperativa (PC) consiste em estimar o valor de variáveis do ambiente de uma forma baseada em medidas disponibilizadas por um grupo de sensores, e em resultado da cooperação entre eles. A percepção cooperativa estende a fusão sensorial i) removendo observações de sensores com avarias; ii) resolvendo problemas de acordo e desacordo entre as observações dos sensores. São estudados 2 algoritmos de PC populares na literatura, Linear Opinion Pool (LOP) e Logarithmic Opinion Pool (LGP) e são apresentados dois algoritmos inovadores que ultrapassam algumas das suas dificuldades: Cooperative Opinion Pool (COP) e P-norm Opinion Pool (POP). O algoritmo COP leva em conta todas as dependências entre observações, de forma similar ao LOP, e reduz a incerteza como o LGP. O algoritmo POP é projectado para satisfazer um conjunto de especificações. Comparado com o LOP e o LGP, o algoritmo POP leva em conta especificações relativas a critérios como a redução de incerteza, tolerância a falhas, actualização sequencial e monotonicidade, e lida com situações de acordo e desacordo entre sensores de forma natural. O desempenho dos 2 novos algoritmos apresentados na tese é testado num ambiente multi-robô simulado e num cenário real, mostrando-se a sua superioridade face aos algoritmos LOP e LGP. A percepção cooperativa activa (PCA) melhora a qualidade da estimativa não apenas através da fusão de medidas múltiplas, mas também através da selecção de acções adequadas de um conjunto potencial, para o subconjunto de sensores activos. Na PCA, define-se uma função de custo tal que a incerteza das estimativas se reduz a um valor mínimo em resultado da selecção de acções óptimas. Um exemplo típico consiste em mover um robô móvel com uma câmara, tendo como objectivo melhorar

a exactidão da localização de um objecto inicialmente observado apenas por câmaras estáticas. Estuda-se também o problema de planeamento do caminho mais informativo para uma equipa de robôs numa rede de sistemas robóticos, particularmente em ambientes urbanos equipados com uma rede de câmaras de vigilância. Um processo de decisão de Markov (PDM) é utilizado para modelar o problema. O conceito de prémio em PDMs é instanciado por uma função de prémio que pesa diferentes objectivos. Um algoritmo com 2 níveis é utilizado: primeiro, os caminhos possíveis são determinados. A seguir, o melhor caminho é seleccionado. O algoritmo proposto é aplicado a simulações e experiências com robôs reais, sendo os seus resultados apresentados e discutidos. Parte deste trabalho experimental foi desenvolvido no âmbito de um projecto europeu.

Palavras-chave: robótica móvel, fusão de sensores, fusão de sensores probabilística, a percepção de cooperação, acordo, desacordo, a percepção cooperativa ativa, em rede robô sistemas, MDP, planejamento de caminho

Abstract

This thesis focuses on problems of cooperative perception and active cooperative perception for networked robot systems.

Cooperative perception (CP) is the problem of estimating the value of environment variables based on measurements provided by a group of sensors and as a result of cooperation among them. Cooperative perception extends sensor fusion by: i) removing observations of faulty sensors; ii) resolving agreement and disagreement issues among the observations. Two popular algorithms for CP are overviewed, Linear Opinion Pool (LOP) and Logarithmic Opinion Pool (LGP), and two novel algorithms to overcome their weaknesses are introduced: Cooperative Opinion Pool (COP) and P-norm Opinion Pool (POP). COP considers all of the dependencies between observations, similarly to LOP, and reduces the uncertainty such as LGP. POP is a specification-driven cooperative perception method. Comparing to LOP and LGP, POP specifically takes into account specifications concerning criteria such as uncertainty reduction, fault tolerance, sequential updating and monotonicity, and naturally, handle situations of agreement and disagreement between sensors. The performance of the two novel methods is checked on a simulated multi robot environment and in a real world scenario, and is successfully compared with the performance of LGP and LOP.

Active cooperative perception (ACP) further improves the estimate quality not only by using multiple measurements but also by selecting appropriate actions from a potential set for the subset of active sensors. In ACP, a cost function is defined such that the uncertainty in state estimation is minimized as a result of selecting optimal actions. A typical example of this consists of moving a mobile robot with a camera to improve the accuracy of locating an object initially observed by static cameras only.

The problem of most-informative-path planning for a team of robots in a networked robot system, in particular in urban environments equipped with a network of surveillance cameras, is also studied in the thesis. A Markov decision process (MDP) framework is used to model the problem. The concept of reward in MDP is instantiated by a reward function that weights different objectives. A two-stage algorithm is used: first, the possible paths are determined. Then, the best path is selected. The proposed algorithm is applied to simulations and real world experiments and its results presented and discussed. Some of the experimental work was accomplished within the framework on an European project.

Key-words: mobile robotics, sensor fusion, probabilistic sensor fusion, cooperative perception, agreement, disagreement, active cooperative perception, networked robot systems, MDP, path planning

Acknowledgements

I would like to deeply thank my advisor professor Pedro Lima who provided me the great opportunity to work, gave me untiring help during my difficult moments and for his encouragement, guidance and great support throughout my study period both at scientific and personal levels. It has been an honor to be his Phd student. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating.

I would also like to express my gratitude towards members of the jury who have accepted to take the time to read and evaluate my thesis and provided their critical comments, encouragement and kindness.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. This work was partly supported by Fundação para a Ciência e a Tecnologia (FCT) grant No. SFRH/BD/23394/2005, and partially supported by FCT ISR/IST pluriannual funding, through the POS-Conhecimento Program that includes FEDER funds.

I would also like to thank the members of IRSGroup and all ISR staff members for their support and friendship over the years.

Contents

List of Abbreviations

ABS	Antilock Braking System
ACP	Active Cooperative Perception
AVA	Active Vision Agents
COP	Cooperative Opinion Pool
CP	Cooperative Perception
CS	central station
DS	Dempster-Shafer
GPS	Global Positioning System
JDL	Joint Directors of Laboratories
KF	Kalman Filter
LGP	Linear Opinion Pool
LOP	Logarithmic Opinion Pool
MCL	Monte Carlo Localization
MDP	Markov Decision Process
MSE	Mean Squared Error
NRS	Networked Robot Systems
OP	Observation Point
PC	Percepção Cooperativa
PCA	Percepção Cooperativa Activa
pdf	probability density function
PDM	Processo de Decisão de Markov

List of Abbreviations

pmf	probability mass function
POMDP	Partially Observable Markov Decision Process
POP	P-norm Opinion Pool
POPT	Plan an Optimal Path for a Team of Robots
SKF	Switching Kalman Filter
UGV	Unmanned Ground Vehicles
URUS	Ubiquitous Networking Robotics in Urban Settings
WLC	Weighted Linear Combination of observation

Chapter 1

Introduction

Autonomous robots depend on information provided by their sensors to interact with the environment. However, sensors in general provide noisy and uncertain information, and might even fail completely in certain situations. Sensor fusion consists of combining the information from several sensors to reduce the inherent uncertainty of information they provide. During the last few decades, this problem has been tackled by researchers from different fields, but it is still considered a hot topic [68, 15, 71, 86, 30, 95].

Sensor fusion models can be categorized based on the type of sensor output models they assume. For instance, a probabilistic sensor model delivers its measurement in the form of a probability density function (pdf). Comparing to classical sensor fusion where the output of a sensor is a value with a given uncertainty, in a probabilistic framework the measurements expresses a degree of belief at each point of the observation space. Hence, probabilistic sensors employ a richer representation of uncertainty, allowing for a more accurate modeling of their expected error. Linear Opinion Pool (LOP) and Logarithmic Opinion Pool (LGP) are two popular probabilistic sensor fusion methods which have been

widely utilized [55, 84, 80, 29, 32, 90]. However as will be shown ahead, LGP method is not fail safe, and LOP does not decrease the uncertainty even if there is complete agreement among the sensors. To overcome the difficulties of LOP and LGP, we introduced two new methods : Cooperative Opinion Pool (COP) and p-norm Opinion Pool (POP). Both of the latter methods consider all of the dependencies between observations, such as LOP, and reduce the uncertainty, such as LGP. However, COP needs more computational power compared to POP. On the other hand, COP is more flexible compared to POP. POP is designed to meet some properties. We check and compare the performance of the CP methods by running different simulations.

On the other hand, fusion of inconsistent observations can be erroneous. In addition, prior to fusion of information, faulty observations should be discovered and removed. In this thesis, we propose the Cooperative Perception (CP) concept under which fusion of information, removing faulty observations and problems of agreement and disagreement among observations will be integrated.

The new generation of sensors, besides measuring the parameter of interest, are able to perform other tasks such as changing its viewpoint and changing the internal and external parameters or recognize the environmental conditions. Some of the sensor actions can reduce the measurement uncertainty. This motivates us to reconsider the estimation problem to determine if selecting adequate actions can improve the estimation quality. For that purpose, we will introduce the Active Cooperation Perception (ACP) framework, where we show how to further improve the quality of estimation.

1.1 Cooperative Perception

Cooperative Perception (CP) is the problem of estimating the value of environmental variables based on measurements provided by a group of agents and as a result of cooperation between them. In cooperative perception the problems of information fusion, removing faulty observations and agreement and disagreement among the observations are integrated under a unique framework. Cooperative perception is a concept broader than sensor fusion. In cooperative perception, in addition to fusion of information sources, we consider the problem of agreement and disagreement and information sharing among the sources. Recognizing and removing faulty observation prior to fusion is vital because a faulty observation may affect the decision of an agent which depends on sensors measurements. Combining the measurements received from several sensors taking into account that some of them might fail and/or disagreement among sensors might occur due to errors, we study methods that can successfully cope with situations of agreement and disagreement between sensors.

1.2 Active Cooperative Perception

Recent sensors, in addition to measuring the quantity or quality of interest, have extra capabilities. Among others, we can refer, e.g., to self calibration, self diagnosis, fault tolerance, network self configuration, reconfiguring network topology upon fault, discover faulty sensors. On the top of that, sensors may be active, i.e., they may change their view point, either because they are assembled on the top of a vehicle, and/or have an associated pan-tilt device. This provides enough motivation to reconsider the estimation problem.

Active Cooperative Perception (ACP) refers to further improvement in the measurement quality not only by using multiple measurements but also by choosing appropriate action for each of the sensors from a potential set. In ACP, the action is not only limited to information fusion and information exchange. A cost function is defined such that the uncertainty in state estimation decreases as a result of its optimization by selecting optimal actions.

ACP problem is closely related to the estimation problem. The major difference is that, in classical estimation theory, the sensors have been considered as passive elements of the recognition system. However, in ACP, the sensors are active and can perform actions. In some situations, the action results might improve the estimation quality and the estimation uncertainty becomes a function of sensor actions.

Therefore we have to find the action set and the sequence of actions in order to optimize the cost function under some constraints. An example of ACP is multiple target tracking where, e.g., a group of mobile sensors (e.g., pan/tilt zoom, cameras on board of mobile robots) and static sensors (e.g., fixed cameras) are tracking a number of targets in which the goal is to estimate the state of a group of moving targets based on uncertain measurements the sensors. However, state estimation uncertainty may be reduced by changing the relative pose of the mobile sensors with respect to the targets. A pan-tilt camera can change its field of view and may provide more certain information about the area where the targets are or where they are expected to move. Changing relative pose, adjusting pan/tilt angles, zooming in/out cameras, turning sensors off/on are examples of actions that might improve the group perception. There might be some members, e.g., fixed cameras, that can not perform any action to improve the observation quality, although the observations provided by those sensors may influence the final decision [3]. We provide a formalism for ACP in

general and also consider some special cases.

1.3 Thesis Contributions

The thesis has three principal contributions:

- Introducing two probabilistic perception methods, Cooperative Opinion Pool (COP) [60]and p-norm Opinion Pool (POP) [62] is the first contribution of the thesis. Both are defined to be applied to probabilistic sensors. COP considers all of the dependencies between observations. POP is a specification-driven cooperative perception method. We define our objective function and derive POP analytically. We also show how POP and COP methods satisfy a series of defined properties. Through experiments we will show that the two methods can remove the erroneous observations and solve the disagreement problem.

- Defining and formalizing the ACP concept is the second contribution of the thesis. We show how much an action of a sensor can improve the estimation quality. Finally, we show how the optimal action set can be chosen.

- Another contribution of the thesis is how to actively plan the paths for a team of robots considering a set of parameters[61, 10]. This set of parameters express the users constraints and is not fixed. The number and priority of the parameters can vary depends on the scenario. As the environment can be highly dynamic, the algorithm should consider the current situation and replan only if necessary. This tells us that the algorithm should be realtime. We show how to use the ACP concept to address the problem of multi-robot multi-goal planning.

Chapter 2

Related work

2.1 Introduction

Cooperative Perception (CP) extends sensor fusion by considering disagreement among the observations, mobile sensors and data exchange among the sensor agents.

An Active Cooperative Perception (ACP) system is able to improve its perception by performing actions. Through the information received from the feedback, the system can reduce amount of observation uncertainty. This process continues till the system reaches an equilibrium point where observation quality can not be further improved, or the cost of improvement is higher than the benefit or it is negligible.

In this chapter we overview the related work in the field of sensor fusion, CP and ACP.

2.2 Sensor Fusion

In the daily life, we use our sensors to perceive the environment. Our five senses help us to build a model of environment and to understand what is happening there. This model

is frequently updated based on new information. This helps humans to interact with the environment, making decisions, planing future actions and refining their knowledge about it. However, such sensors can not accurately cover the whole world individually.
For example, each of our sight sensors are built for 2D interpretation of the world, but by fusing information from the two eyes, humans obtain a 3D view of the world. In fact, using our interpretation capability, by fusing two similar sensors, a new form of information can be created. In some situations, information of different sensors is collected in order to increase the information quality. For example touch sensors can improve the quality of sight sensor measurements.

Sensor fusion has been an active area in the last few decades. The main reasons are emerging new application such as sensor network, increasing processing power, improving wireless communication systems, and the need to cover wider areas.
The process of combining information of several sensors is called sensor fusion. In general, the quality of fused information is expected to be better than each individual measurement. An extensive research has been done on sensor fusion during the last few decades by scientists and researchers from different fields. Among others we can refer to medical applications, robotic applications, image processing applications, military applications and bio-medical applications. To overview the works, we categorize the fusion algorithm and then review some works from each category. However, this categorization is not unique and there are many ways to categorize sensor fusion algorithms.

One of most the popular method for categorization of sensor fusion algorithms is JDL [40]. JDL is an acronym for the Joint Directors of Laboratories (JDL) Data Fusion Working Group which was established in 1986 to unify the terminology for data fusion. The JDL model is depicted in Fig 2.1. This model is very general and useful. By using it, one

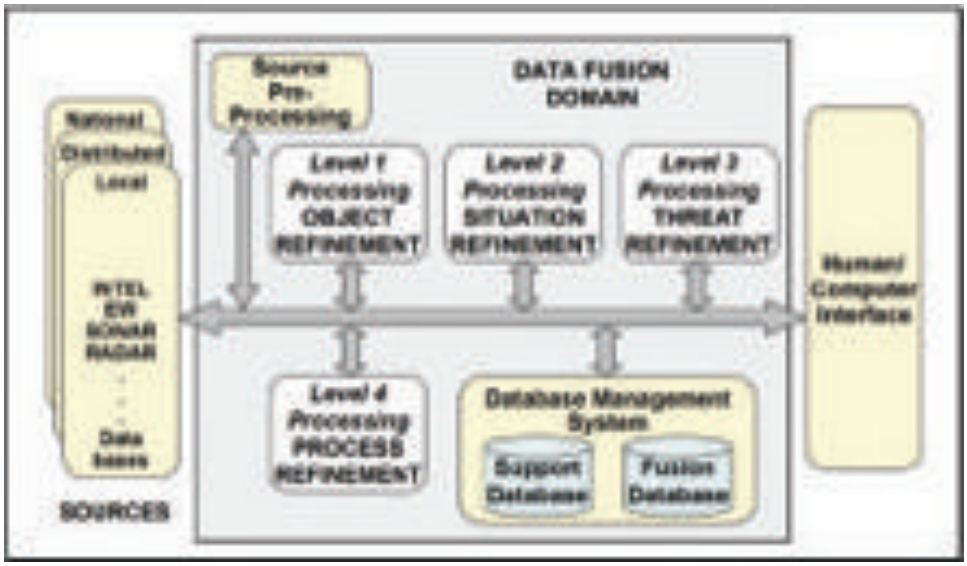


Figure 2.1: *The JDL data fusion model. JDL/DFG data fusion model divide the various processes into 4 levels: Level 1: Object refinement, Level 2: Situation refinement, Level 3: Impact Assessment (or Threat Refinement) and Level 4: Process Refinement. JDL model is useful for visualizing the data fusion process and also for facilitating discussion and common understanding. The figure is taken from[52]*

can identify the processes, functions and other characteristics of data fusion method. In this model, a sensor fusion algorithm is categorized into six levels based on the processing stage of information.

A more abstract categorization was used by some researchers which has the same basic principles but only considers three levels: Low Level, Mid Level and High Level.

In the Low Level fusion [87] the sensor data is used for the fusion. By sensor data we mean the data directly received from the sensor without any further processing. This does not include the primary signal processing such as analog to digital conversion, noise removal and so on. This level of fusion is appropriate if someone wants to have the most reliable and accurate estimation of an unknown parameter. For example, in applications such as localization or measuring velocity, the fusion is done at this level. Probabilistic methods and machine learning methods can be used to fuse the data at the low level.

In the Middle Level fusion [28], features extracted from lower levels are used to perform data fusion, e.g., edges, gray levels and contours. Actually low level data is used to generate these features. We can use probabilistic algorithms and/or learning algorithms for fusion of the information at this level.

In High Level fusion [53], experts combine information obtained based on lower level information. This level is also called decision-making level. For that, each expert builds its own interpretation and then, based on the other experts' information, a final decision is made. By doing that, the confidence level of decision making can be increased. This is possible especially if the type of information is complementary. In this area we can refer to probabilistic methods, possibilistic methods and statistical methods as popular methods of data fusion at this level.

It is worth to mention that this categorization does not cover all the existing fusion ap-

proaches. It is possible to fuse signals from different levels. One way to categorize the fusion methods is to consider the form of data we receive. We can refer to probabilistic versus possibilistic data as two different data formats. In the probabilistic approach, a probability is assigned to each point in the state space while in the possibilistic approach things are a bit different. In possibilistic approach, we allocate a probability mass to sets or intervals.

Barberá et al [8] studied a scenario in which a robot navigating in unknown and unstructured environments performs delivery-like tasks. The robot was equipped with two uncertain sensors: sonar and infrared. The goal was to reduce the uncertainty of the measurements. A fuzzy logic [99, 100] algorithm was used to fuse the data and obtain more reliable data. They developed a validation environment including models of sonar and infrared sensors, addressed the sensor fusion problem using fuzzy rules and automated the fuzzy rules training process. In the experiment, the method is compared to a benchmark method. The results show reductions in average error and variance and were very encouraging.

Another approach for possibilistic fusion of the data is the Dempster-Shafer (DS) [76] approach which is similar to the fuzzy approach. The Dempster-Shafer theory of evidence has uncertainty management and inference mechanisms analogous to our human reasoning process.

Since sensor data is not in the form of possibility, the data should be converted. After that, Dempster rule of combination [58] is used to combine the data.

Sensor fusion by learning is one way to fuse data of several sensors. One important issue in sensor fusion is having access to the sensors model. Unfortunately, building the model of a sensor is not an easy task. This becomes more difficult if we deal with a number of heterogeneous sensors. Because of that, most often, an approximated model is considered.

Consider a simple example where we are about to fuse information of a robot in order to localize it. One way is to put the robot in a known state and at the same time measure the sensors output. After putting the robot in all the possible states and measuring the sensor output, we have a mapping from real space to measurement space. Then we have to find the reverse mapping, from measurement space into the real world. This is a time consuming task. Instead we can use neural networks [44] for learning the model of the sensors. They can also be used for fusing the information of many resources.

Neural networks are also considered as a good tool for learning a fusion process. Most neural networks have three stages. From raw data some features are extracted. The input layer is the first layer. Depending on the processing needed for the raw data, the input layer has different structure. The output layer is the last stage. In this stage the classification occurs and the final decision is made. Depending on the number of classes and the coding, the number of outputs are chosen. The middle stage is the processing unit where the input features are mapped.

Fukuda et al[41] used a combination of fuzzy logic and neural networks to fuse the information of an array of sensors for curved metal surface cutting robot system. This system cuts products using the data of works' surface shape which are obtained by the measurement. High accuracy sensors are used for precise cutting. In order to use these sensors effectively, a multi-sensor integration system was used based on the neural networks and fuzzy inference techniques. In the experiment 10 sensors were used. Through extensive experiments, the approach was found effective.

Thrun [89] suggested to use two neural networks with one hidden layer and eight hidden units: the sensor interpretation network and the confidence estimation network. The former was used for estimating the occupancy value. The latter was used for predicting the uncer-

tainty of the first network. The networks represent both the expected reward and the confidence in these expectations. Although the results were promising, the author concluded that an important limitation of neural networks for robotics applications is the difficulty in training the neural networks by supervision in an known environment. If the number of states is huge, training the network will be a difficult and time consuming task. This becomes more serious if we have a number of cooperative robots.

Woo-Kyung et al [17] used complementary neural networks for ultrasonic sensor fusion and radial basis function network (RBFN) for environment sensor fusion for a snake robot. The snake robot was equipped with gas and temperature sensors to measure the degree of environmental hazard. The fusion unit uses the information of the robot's sensors and the environmental sensors and relate a value to the current position of the robot. The value represents the hazard degree of the robot's position. A competitive neural network was chosen over a general multi-layer neural network. Competitive neural networks use unsupervised learning and classify input pattern without given information about the target value. The result of fusion was reported successful.

Cherif Smaili et al [78] employed Dynamic Bayesian Networks for precise vehicle localization and road matching. In this work, a multi-sensor fusion strategy for a novel road-matching method was designed to support real-time navigational features within advanced driving-assistance systems. By fusing data from Antilock Braking System (ABS) sensors, a differential Global Positioning System (GPS) receiver and an accurate digital roadmap, driving-assistance systems were implemented. The multisensor fusion of GPS and odometry was accomplished by a switching Kalman filter (SKF). Flexibility and modularity were referred as two important characteristics of their proposed method. The authors claimed that adding new sensors increased the level of robustness. Experimental results showed

that the performance of the system improved especially in ambiguous situations.

When the output of the sensors are in the form of a probability mass function, a probabilistic approach can be employed to fuse the information.

In [29], Durrant-Whyte describes a general framework for modeling senors and the information they provide. He considers a multi-sensor system as a team of decision-makers. Each sensor plays the role of a decision maker. The group of sensors are teamed and have a common goal. Each sensor is able to make local decisions and also participate in team decision. The sensors have ability to communicate and exchange the information. Each member can negotiate with other team members. Three models are considered to explain the behaviors of the sensors: observation model, dependency model and state model. The observation model explains the relationship between the measurement and the reality. The state model explains the effect of robot's position and state on the robots observations. Sensors use a dependency model to combine information if certain conditions are met. However this establishes a dependency between the sensors. In this work, he addresses the problem of cooperation between the sensors as well as competition between them. Finally, an example of the framework applied to a group of independent sensors whose outputs are in the form of an unbiased normal distribution is explained. One issue which is not addressed in [29] is how to solve consensus problem when there are disagreements among team members. In a team, team members should reach an agreement in order to follow an unique policy.

Some researchers divide the probabilistic sensor fusion approach into two categories: Bayesian and non-Bayesian. In Bayesian approaches, having fused observations of $N-1$ sensors, the observation of an N^{th} sensor can be fused with the remaining observations by using Bayes' formula. In non-Bayesian approaches, this sequential fusion of the information is

not possible.

A Bayesian approach for combining probabilistic information of sensors is Logarithmic Opinion Pool (LGP). Elfes [32] used LGP to fuse the observations of several sensors. He assigned a probability value to each cell of the occupancy grid map. This value represents the probability of occupying the cell by an object. He used LGP to combine occupancy grids derived separately for two sensor systems. Considering that LGP is a Bayesian approach, the occupancy grid map was incrementally updated by fusion of information of several sources. In the experiment, the data from two sensor systems: a sonar sensor array and a single-scanline stereo module that generates horizontal depth profiles were fused together.

Solberg et al [79] employed LGP approach for multisensor image classification. In this work, different weighted LGP methods were used for fusion of decision for classification of the images. Finally, the performance of the method was examined using three databases. They concluded that using equal weights delivers the best performance. The performance was relatively robust to a range of weight values.

Stroupe et al [82]used Bayes rule and Kalman filtering to fuse data. They implemented a real-time sensor data fusion on a reactive multi-robot system for several different applications. The method was tested in three application areas, including object localization, object tracking, and ball position estimation for robotic soccer. By fusion of information they tried to improve accuracy of estimation. Employing the approach, it was empirically demonstrated that more observers achieve a more accurate estimation.

Linear Opinion Pool(LOP) is a non-Bayesian approach for fusion of information. V. Rigaud et al in [67] exploit LOP for fusion of information. In this work the goal was to estimate an optimal location by fusing data driven from log and gyroscope or a non-optimal location

by composition of compass, depth-meter, and log data. A positive weight was assigned to each information source. This weight reflected the probability that the robot was in a particular cell for a given element of the weighted vectorial model of one acoustic measurement. Then, using LOP, they fused the information of several sources. They claimed that although in total the estimation was suboptimal, they could reduce the computational capacity needed and the memory demand.

Sun et al [84] used LOP to fuse data collected for vehicle reidentification. In this application, a vehicle is identified in order to extract different parameters such as travel time, travel time variability, section density, etc. They fused information of inductive and video data for identifying the vehicle. The weights are determined by searching an n-dimensional grid of real numbers and finding the optimum combination that gives the best performance on the training data alone. In total 6 features were used. Their results have shown that fusion yields better results than the use of a single detector. LOP is also employed by Soong et al [80] for speaker recognition, Miao et al [56] for mine detection, Benediktsson et al [11] for fusing the output of several Neural Networks.

2.3 Cooperative Perception

Cooperative perception is a concept broader than sensor fusion. In cooperative perception, besides the fusion of information from different sensors, we use the information of other agents to check for outliers, disagreement and missing information. Each agent fuses the information received from its sensors, then it broadcasts that information and makes it available to the other agents. Many researchers [91, 69, 26, 83] used sensor fusion to improve the observation quality and handle the limitation of sensors by aggregating the

information received from different sources. However, most of the researchers do not consider the problem of disagreement among the observations. This disagreement can cause problems for a team of robots (e.g., a wrong decision can be made or a wrong estimation can be made).

Tischler [91] defines cooperative perception as exchange of information among a number of agents through a reliable communication channel. In this work, multiple vehicles equipped with an inter-vehicle communication device exchange their position (the vehicles poses are provided by GPS and video information and other objects around provided by vehicles cameras, e.g. pedestrians) in a joint coordinate system. It includes the position of the cars and other detected objects. Since the field of view of each car is limited by either its sensors field of view or occluded by natural obstruction or other vehicles, inter-vehicle data exchange can extend the field of view of the agent and observable domain to inaccessible regions. Each vehicle acts as sensor and fuses its information with the perception of others to obtain an enhanced environmental description. A Kalman filter is used to fuse and update the asynchronous information received from different sources (GPS system, video cameras and digital map). To do so a time stamp is assigned to each piece of information . In the selected traffic scenario, the tracking algorithm has shown the desired output (an output with error defined in a optional bound) of the cooperative perception. In this work the problem of disagreement among sources of information is not considered and the method is not tested in the presence of outliers or disagreement.

While perception of a robot is restricted locally, due to limitation of the number of robot sensors and the view field of each sensor, by sharing information a more general view of the environment can be provided [48]. In this work, information extracted from the cameras and infrared line sensors is used to track humans and robots. A cooperative perception

system is established based on robots and environmental sensors. The information is used to guide the robots to a specified position. The cooperative perception can reduce the number of required sensors. Several experimental results of the cooperative perception in the navigation of a partner robot is performed which supports the effectiveness of the proposed method. The effects of sensor failure and the disagreement among the environmental sensors are not considered.

In [75] cooperative perception is used to effectively and accurately estimate the state of environment for a team of robots in presence of uncertainty. In this work, a decentralized approach is used that balances use of communication bandwidth with the need for a good assessment. Establishing cooperation among robots can extend the perception and reduce the information uncertainty. However, it has a cost: cost of communicating data. Therefore, a utility function is defined in which the improving the perception is considered as reward and the communication is considered as the cost. In this work the problem of disagreement among the agents is considered. A hard challenge can occur when there is disagreement among the agents. To solve this, the information has to be passed back through all the robot chain, until it reaches the proposal initiator. However, there is no guarantee that the team can reach an agreement. Another point is that a team either can reach an agreement or not but there is no partial agreement.

In [26] cooperative perception is used to gain a global view of the world. The application domain is the RoboCup middle size league in which robots are prohibited to have a direct global view. However, a global view can be created using the information provided by the teammates through communication. Two different kinds of tracking scenarios are considered. In the first scenario, it is assumed that the number of objects to be tracked is unknown, sensor data is noisy and reliable. In the second scenario, only one object is

being tracked and sensor data is noisy and unreliable. It is shown that when there is unreliable data which is constantly provided by faulty sensor, the Kalman filter can fail. To solve this, a combination of Kalman filter and Markov Localization is used to remove the outliers. However, the cases when the number of outliers increases or when there is partial disagreement are not considered.

2.4 Active Cooperative Perception

Here we first overview ACP work in the robotic domain and then a number of similar works will be examined. Although a limited number of ACP approaches by robotic researchers has been found in the literature, there is a considerable amount of work carried out by other researchers.

Fox et al studied the problem of Active Cooperative Localization in [37]. In this work, an active localization approach was introduced. Robot's motion direction is controlled to improve the quality of localization. In addition, the sensor orientations are panned so as to achieve the best observation of the robot which is called active sensing. In other words, the problem of active sensing and active navigation were studied under an unique framework; two problems were simultaneously optimized. The framework used for localization was Markov localization[38] which is a passive probabilistic approach for localization. The main objective was to minimize future expected uncertainty considering all costs (e.g., the cost imposed by implementation of actions). Entropy was considered as a measure of uncertainty. The lower the entropy the more certain the estimation. Initially, the sensor models were calculated and stored. The environment was a 2D environment and therefore for each pos of the robot, the sensor model is determined and memorized in a lookup table.

To obtain the most accurate robot localization, a utility function and a cost function were defined. Utility of an action was related to the uncertainty of the localization belief. Cost of an action was defined based on the energy consumed to perform the action plus the time needed to perform the action. An action selection objective function was also defined based on these two. In the experiment a robot was placed in a symmetrical corridor. The only way to localize the robot was to move the robot to the rooms. The results of experiment show a noticeable improvement in uncertainty and error reduction. It was pointed out that the results could be improved if there were more distinguishable features in the environment. In this work the authors only considered a few number of the states for the robot. Also, the only parameter that affects the robot path is the localization uncertainty. The problem of multi robot path planning is not addressed.

Ukita and Matsuyama in [93] presented a real-time cooperative multi-target tracking system. In their configuration, a number of active camera which were networked cooperatively tracked a number of targets. The cooperation was established via dynamic exchange of information about the targets. As a result of the cooperation, the system could accurately track multiple targets in real time. Two modules were defined: perception and action modules. Information flows between the perception and action modules. The former obtains the current camera parameters from the latter to generate the background image and the latter receives the current target location from the former to control the camera. Therefore in one time step the system perceives the environment and in the next time step it implements the action. In order to compensate the lag of performing the action, a new dynamic system architecture named as dynamic memory architecture was proposed in which the visual perception and camera action modules run in parallel and dynamically exchange information via a specialized shared memory named the dynamic memory. The algorithm was applied

to a scenario in which 10 cameras were installed in a room to track a group of humans. The best action is determined based on the current position of the targets. A multi model can help to determine longer term rewards.

Arskino and Ribeiro in [4] proposed an active range sensing for mobile robot localization. In their approach natural landmarks were chosen in a such way that the highest update rate of location estimation could be reached and the uncertainty in localization was minimized. This goal was attained by actively and continuously perceiving small regions of the surrounding environment to reason about the expected and the acquired information on landmarks. A series of constraints on landmark visibility were defined. Then cost for each landmark was calculated. Then the objective function was optimized which resulted in a Kalman filter optimization. Also to improve the localization accuracy and robustness, laser range data and odometry were fused using Kalman filtering. The planned path is not optimal as it is chosen based the surrounding landmarks. Moreover, it is not shown how to integrate other parameters such as the path costs.

Davison and Kita [21] presented an approach for 3D SLAM using active vision. In this work the type of features in the map for each time instant were chosen by considering two criteria: expected visibility and the information content of the measurement. However, the expected utility is not considered for an infinite horizon. After identifying the set of available features, the amount of information gain for each was calculated and the one selected was the one which had the highest innovation covariance. In this work, integration of other parameters is not considered.

Cassandra [16] used a Partially Observable Markov Decision Process (POMDP) framework to find an optimal solution for mobile robot navigation. One of objective of the work was to choose an action that decreases the uncertainty the most. Achieving the goal was

another objective. These two concerns were put in an one unified framework. Expected entropy was used to measure the uncertainty. After implementing the robot's action, the objective function is minimized to find the best action. The entropy is used to measure the uncertainty of the beliefs. When the robot is confused, the entropy of the belief is high and we should choose an action that reduces the entropy. To deal with the computational cost of POMDP a heuristic search was used. Although the algorithm becomes faster, it might not be the optimal solution. It is also not shown how to consider other possible factors the may affect the robot path. The control strategy algorithm was examined both in simulation and in runs on a robot. Three separate sets of experiments with different configurations in synthetic environments were performed. Different methods for approximation were tested. Although most of the approaches did well, in some cases they failed. A real scenario with 1052 POMDP states was considered. The results were consistent with the simulation. When the initial state was not specified, the goal might not be reached. However, the abilities of the POMDP models to control active perception were not considered in this work. Another drawback of the approach was that it would become quite expensive to compute if the expected entropy resulting for a long sequences of actions was considered. So the control strategy was considered for a short range of actions. The performance of the robot decreases rapidly if a major change happens as a result of considering a static world. Although the framework can cope with a dynamic world, it can only deal with small changes. A hierarchical agent-based active vision approach for the dynamic coordinated selection and positioning of active-vision cameras for the simultaneous surveillance of multiple objects-of-interest is presented in[5]. Trajectory of intruders was unknown and the environment was considered cluttered. The set of actions includes change in angles and positions of the cameras. Both simulation and real world experiments showed the effectiveness of

the algorithm. In this approach idle cameras were used in anticipation of future service requirements. Therefore not only the current configuration was considered for the optimization, the future positions of intruders were also considered. The algorithm proposed in the work dynamically adjusts the orientation and position of the cameras in order to maximize the system performance by avoiding occlusions and acquiring images with preferred viewing angles. A parameter named as visibility measure was defined which represents the amount of information that a camera can deliver. Three type of agents were defined: sensor agent, referee agents and judge agent. For each sensor, a sensor agent was considered which has the duty of choosing which target will be serviced at every demand instant by the associated sensor and determining its optimum pose in order to maximize the sensor performance metric (i.e., visibility) over the span of the rolling horizon. The referee agent monitors the intentions of all sensor agents and ensures that external rules are not violated. A positioning strategy aims was defined to find the optimal position of sensors considering other parameters such as current sensors position or motion capability. Simulation results and a real world experiment the effectiveness of the algorithm.

A real-time approach for multi-target tracking by communicating active-vision agents was introduced in [94]. In their scenario, a number of Active Vision Agents (AVA) were considered. A camera was assigned to each agent. The goal was to track some targets cooperatively. The system can change its behavior in an adaptive manner which depends on the situation and the given task. It can keep tracking and focusing on some objects of interest. The agents were networked and dynamically exchange the data. It was claimed that their algorithm could track multiple moving objects simultaneously under complicated dynamic situations in the real world scenario. In the algorithm the active agents simultaneously monitor the environment and when an intruder is found, the information is shared with the

rest of agents. The agents capture the images asynchronously and therefore, the system is not in need of any synchronization mechanism. Each AVA works autonomously while maintaining its own intrinsic dynamics and cooperates with the other AVAs by exchanging information through the network. To cover a wider area and simultaneously acquire more detailed information (e.g., for object recognition tasks) with a limited number of ACP, controlling pan/tilt/zoom of cameras were employed. A decentralized architecture is considered in which each agent individually decides if it has to participate in the surveillance of the intruder. To implement real-time cooperation among AVAs, a three-layered interaction architecture was designed. In each layer, parallel processes mutually exchange a variety of information for an effective cooperation. To verify the effectiveness of the task representation with the task-constraint and the object-priority, two experiments were performed in the same environment. Experimental results demonstrated that the proposed real-time cooperation method enables a system with ten AVAs to successfully acquire mobile object information and adaptively assign an appropriate role to each AVA. However, they did not consider any uncertainty.

Denzler published a series of works in the context of active vision. In [23], he used an active vision framework to classify a series of objects. In this work, entropy was used to measure information content. The mutual information which is defined as the difference between information content before and after implementing an action was used to measure the information gain. At first a function which is a mapping from state to action was learned. In an ACP framework, it is necessary to know in each state which action should be implemented. For that, in each state, the mutual information is measured for all possible actions. After having them all, we choose the action that provides the maximum information gain. After learning the function, having the prior belief and the observation

probability distribution, the best action is chosen based on the value of mutual information. In this work, changes in camera angles were considered as the set of possible actions and a sequential decision process is used to update the prior belief. The prior belief was updated based on the observation acquired after implementing each action. For the experiment, an object recognition scenario was designed. A pan/tilt/zoom camera was used to take the images from different views. During the sequential decision process, the camera looked to parts of the object that allowed the most reliable distinction of similar looking objects. The experiments were performed with discrete density representations as well as with continuous densities and Monte Carlo evaluation of the mutual information. The results showed that the sequential decision process outperformed a random gaze control, both in the sense of recognition rate and number of views necessary to return a decision. In this work only a camera was considered. Moreover, the object of interest was stationary. Also there was enough time to perform the action. However in a dynamic environment where an object of interest is moving and there are mobile obstacles, this methodology may have troubles. The first trouble is that the size of state space becomes too large. Also there may not be enough time to perform a sequence of actions.

In [22], another formalism for active object recognition was proposed. In this work, selection of additional views was considered as an optimization problem, and the optimal sequence of views was learned autonomously. It was shown how to use reinforcement learning for viewpoint training and selection in continuous state spaces without the user interaction. The whole process was considered as a closed loop between estimated state and selected action. After performing the action, new estimates and the reward were calculated. Actually it is the classifier that returns the reward, which measures the quality of the chosen viewpoint. For a viewpoint that increases the information observed so far, the reward

should have a large value. Entropy is used to measure informational content. A method based on particle filter for fusion of multiple views based on recursive density propagation was presented in [25]. To use reinforcement learning a two-steps approach was used. First, estimate the value function during training. Second, at any time the sensor fusion returns the estimated state as classification result and select that camera movement which maximizes the expected accumulated and weighted rewards. Also a way for extending the algorithms for continuous reinforcement learning was introduced. The value function was defined as a weighted sum of the action-values of all previously collected state/action pairs. The experimental results showed that the viewpoint selection was able to select a minimal number of views and perform an optimal object recognition with respect to the classification.

A rigorous mathematical framework for achieve tracking was present in [24]. The main goal was to provide a framework for actively controlling the focal lengths of a binocular camera while tracking a moving object. A general schema for nonlinear and non-Gausssian distribution was presented. An optimality criterion for the case of arbitrary distributed state vectors were derived. To have an analytical form and use optimality criterion of Kalman filtering some assumptions were made. Finally, for Gaussian distributed state vectors, an equation in closed form was derived which can be used in real time scenarios. It was concluded that the action that minimizes the posteriori covariance of error should be selected. However, this should be known in advance. The approach was verified and tested in real time experiments for binocular object tracking. Active focal length control yields a reduction in the estimation error up to 43% compared to tracking in which the focal length was set constant.

Cook et al [20] presented an approach for decision-theoretic cooperative Sensor Planning

between multiple autonomous vehicles executing a military mission. During the mission, an intelligent cooperative reasoning was used to select optimal camera actions in such a way that it maximizes the expected value of information obtained by the sensors and at the same time should minimize information obtainable by the enemy. As there were a large number of vehicles involved in the scenario, establishing cooperation between them might maximize the information gain. Utility of sensing action and utility of maintaining stealth were defined. Based on these two utilities, the global function that the agent was attempting to maximize was defined. An algorithm named as Observation Point (OP) refinement algorithm was used to select optimal observation points from which vehicles could observe a specified area of interest. OP was provided with polygonal descriptions of the area of interest and the selected regions where the vehicles perform observation. OP returns a sorted list of observation point sets, where each set is rated by its visibility/stealth measure. The basic principle of the algorithm is as follows: when an observation point is selected, an agent should trade off the expected value of information acquired on one hand, and the dangers of revealing the agent's presence, on the other hand. Then the utility function is used to rate the suitability of candidate vehicle's locations, given an area of interest to be scanned, and given the suite of sensors on-board of the vehicle. As it was mentioned before, there were a number of vehicles involved in the scenario to increase the chance of a successful mission the robustness level, security, and number of observation points for scouting an area. A combination of central control, local control and distributed control was used. The weighting of each of the four selection factors is controlled by a central force: the mission leader. The selection of each field of view and the dynamic updating of weights and view widths is performed at a local level. For those sorts of tasks a distributed control was used: target confirmation, security hand off, and health checks. The observation point refinement

algorithm was used to select observation points for a set of vehicles that optimizes the co-operative viewing of an area of interest while minimizing the risk of being viewed from the area of interest. To decrease the computational cost, an uniform sampling strategy over the observation regions was used. Schiele and Crowley [74] used the similarities between the active object recognition problem and transmission of information through a channel to formalize ACP. Based on the analogy between the transmission of information through a channel and object recognition, the transinformation of the recognition process can be calculated. Similar to the context the transmission of information through a channel where the relation between the input symbols and the output symbols is measured by entropy, it is also used to measure the gain of information in active vision systems. To determine the most discriminative viewpoints of an object, a second analogy between these two was used. Based on such viewpoints, the authors defined an active recognition algorithm which was able to resolve ambiguities in a single-view recognition approach. The most significant viewpoints of an object was defined the one that maximize the difference between the information content. The results of the experiments showed the applicability of the transinformation for active vision.

Borotschnig[14] proposed an on-line active vision framework for object recognition. The objective in this work was to find the best viewpoint such that the object can be classified with the highest possible uncertainty. In the algorithm, after detecting an object, the first series of hypothesis is generated. If there is any ambiguity, a series of action is planned. After receiving the new information and fusing it with the initial information, again the object is classified. If there is still a problem in classification, a new series of actions will be planned. An eigenspace object recognition was used. For fusion of information a new approach for fusion in the context of eigenspace was proposed. Entropy is used as a measure

of uncertainty. To measure information gain after implementing the action, the difference between the entropies is calculated. The best viewpoint is the one that maximizes the information gain.

Spaan in [81] defines Cooperative Active Perception "as an agent considers the effects of its actions on its sensors, and in particular it tries to improve their performance". The work focuses on the cooperation between a set of surveillance cameras and mobile robots. The robot's sensor and ceiling mounted fixed cameras can provide complementary and redundant information. Both can enhance the global view of the world. A POMDP is used to model the interaction of an active sensor with its environment. To decrease the POMDP computational cost, an approximated solution is chosen. An advantage of taking a decision-theoretic approach using POMDPs is the natural integration of measuring task performance and situational awareness. However, solving the problem using POMDP with large number of states can be too complex for real time applications.

The algorithm we propose is different than the works mentioned above in some ways. First of all, most of the researchers only consider the localization uncertainty as the only factor that influences the path. They do not consider other important parameter in a dynamic environment such as the path length (energy consumed) and maneuvering issues. As a result, integration of other parameters is not considered. Secondly, in most works, only one step ahead reward is considered but not an infinite horizon. While we propose to use a Markov Decision Process (MDP) in our algorithm because of its lower computational cost than POMDPs and optimality, others either use a suboptimal method (heuristic search) or an approximation of an optimal method with limited states.

Grocholsky [43] in his Phd thesis performed research on cooperation between air and ground vehicles to cover large areas searching for targets. In his research a target on the

ground is localized using a single UAV with multiple small UGVs. Sensors on UAVs are considered typically limited in their accuracy of localization of targets. On the other hand, unmanned ground vehicles (UGV) can be deployed to accurately locate ground targets, but they have the disadvantage of not being able to move rapidly or see through such obstacles as buildings or fences. The keys to this are our framework and algorithms for search and localization, which are easily scalable to large numbers of UAVs and UGVs and are transparent to the specificity of individual platforms. They describe how they can exploit this synergy by creating a seamless network of UAVs and UGVs. The keys to this are framework and algorithms for search and localization, which are easily scalable to large numbers of UAVs and UGVs and are transparent to the specificity of individual platforms. A consistent framework is developed for the design of multi-sensor cooperative teams. Information-theoretic utility measures and an established decentralized data fusion architecture are combined with the team decision problem formulation. This results in a decentralized cooperative control architecture for autonomous multi- sensor information gathering systems. There are similarities and differences between Grocholsky's work and our work. Grocholsky used Nash bargaining problem to solve the decision making problem while we use MDP to solve the problem. He used information filter to fuse information from scalable decentralized data sources but we use POP method (Section 3.5.2) to fuse the information from different sources. POP filters faulty information out automatically. However, the information filter does not filter the faulty observations. In both works a utility is defined in which costs and rewards are considered. Both methods are considered a decentralized structure in which the agents consider both its own and team's benefits. In our approach we check and control the data for inconsistency while in Chrocholsky's work they do not consider the inconsistency.

2.4.1 Multi-robot path planning

The problem of multi-robot path planning has been studied by many researchers[66, 98, 18, 63, 92]. In general, the problem is considered as an optimization problem. An objective function is defined and the solution to the optimization problem provides the optimal paths. The objective function is defined based on a set of criteria. Example of criteria are total path lengths and time to reach the goals. Basically, additional parameters can incur more complexity.

Multi-robot multi-goal planning approaches can be categorized based on the amount of information each robot has about the other robots and goals. The methods can be evaluated based on the three basic properties: optimality, completeness and computational complexity. In centralized (coupled) approaches an agent (one of team members or a separate agent) has access to information of other agents (robots and goals) and the planner can find the optimal solution at the cost of high computational complexity. One way to address the centralized approach is to generate all possible paths for each team member and then choose the optimal solution[92]. However, as we mentioned before, this is computationally expensive and for a real-time application might be intractable, especially when the teams size and/or number of goals grows. To solve this, decoupled approaches are introduced. By decoupling the problem, the original problem is broken into several sub-problems, the problem of scalability can be solved and planning becomes faster. However, the methods are suboptimal and incomplete. As a result of incompleteness, a high rate of planning failure can occur which is very problematic in real world applications[72]. In [72], the authors implemented and compared coupled and decoupled planning algorithm. They found that using a decoupled method can increase the failure rate up to 70%. They could speed up

a coupled algorithm by ignoring the conflicts in the generated paths unless it is necessary. A coupled parametric planning strategy for multiple coordination robots based on the task negotiation is proposed in[51]. The method is modular and provides a flexible parametric cooperation. The strategy improves the flexibility of multiple robots. [66] uses a modified pulse-coupled Neural Network for real-time collision-free path planning of mobile robots in nonstationary environments. It works in dynamic environments and requires no prior knowledge of target or barrier movements. Sampling-based planners [18, 46] sample the free space in order to generate curves that represent collision-free paths in the space. However, the problem with these methods is that they are prone to failure[57]. One way to decouple the planning is to consider only part of configuration space for each robot. By doing so, the robots placed close together establish a subteam. The teammates can effect other subteam member's path. On the other hand, the robots placed far away and are in different subteams and have not interaction. The planning is carried out into two phases. In the first phase an optimal plan for each robot is determined and in the second phase collisions are checked[6, 50]. Reactive multi-robot planning methods are suitable for real time applications[47, 42]. However, the main problem is the local optima.

Due to nature of our problem in which the central station has no access to information of all the robots and the goals, a coupled approach is chosen. However, to reduce the computational burden, a two-stage solution is provided. In the first stage, all possible paths are determined and in the second stage the generated plans are evaluated. An MDP is used to find an optimal solution in both stages.

Chapter 3

Cooperative Perception

3.1 Introduction

Similar to human beings, mobile robots use sensors to understand and interact with the environment. To accomplish their tasks, they must be provided with accurate information about themselves and also the environment. Because of that, they are equipped with a number of sensors. However, due to sensor uncertainty, the measurements are not completely accurate and certain. Estimates, conclusions and decisions that are based on those measurements should consider the presence of these uncertainties. Extra sources may help to reduce the effect of those factors and provide a more accurate estimate of the parameter.

In this chapter, firstly, we explain our motivation for studying cooperative perception (CP). Next, two popular probabilistic cooperative perception methods will be overviewed. After that, we introduce two probabilistic methods for cooperative perception. We also conduct a number of simulations to evaluate the quality of the methods.

3.2 Motivation

As we explained before, it is often required that several sensors cooperate. Here, we explain briefly the cost and benefits of cooperative perception. Cooperative perception adds to sensor fusion removing faulty observations as well as, resolving agreement and disagreement among the observations. These can be considered as benefits of cooperative perception.
Cooperative perception has costs. Depends on the applications, different costs can be considered. The followings are examples of cooperative perception costs:

1. For cooperative perception, agents need to exchange information and perform computation.

2. Every cooperative member should have enough knowledge about the cooperative form of perception to know its own commitments towards the other agents. They should be aware of objectives, advantages and disadvantages.

Therefore we need extra energy, communication and computational resources and we consider them as cost.
Based on the above explanation, cooperative perception is a costly operation. There are questions that come to mind when we talk about cooperative perception: why cooperative perception? What justifies a call for the effort to implement cooperative perception in the face of restricted resources? What supports the idea of spending precious resources on cooperative perception? Why should agents be committed to learn about cooperatives perception? Here, we explain some general reasons for that:

1. In many situations sensors are considered to be accurate elements while in reality

their measurements inevitably contain uncertainty and error. Sensor readings are noisy and uncertain because they have limited accuracy, precision, modeling restrictions, stochastic hardware, software and environment noise that affects their output. So, the measurements may not be completely trustable. In other words, there is ambiguity about the measurement which is expressed by assigning an uncertainty to it. Therefore it is necessary to effectively manage and reduce/eliminate the ambiguity. There are faulty sensors that generate faulty observations. Cooperative perception methods can detect the faulty observations. Moreover, handling disagreement is another benefit of the cooperative perception.

2. Generally speaking, in complex dynamic environments, sensors have different partial fields of view. This is because any sensor has a limited range and field of view. The agents at each time instant can only observe part of the environment. However, it is necessary for an agent to have complete information about the environment to take an appropriate decision.

3.3 Methods of Cooperative Perception

There are different ways for categorization of sensor cooperative perception methods. One important factor is the format of the sensor output model. In this work, we categorize the cooperative perception methods based on the sensor output models. Traditionally, the output of a sensor is a *value* with a given uncertainty. One of the most popular point estimation algorithms is weighted linear combination of sensor measurements(WLC)[7]. For

N observations with the estimations $x_1, x_2, ..., x_N$ and variances $\sigma_1, \sigma_2, ..., \sigma_N$ respectively, the weighted linear combination of sensor measurements is determined as:

$$X_{wlc} = W_{wlc}^T X^T \tag{3.1}$$

$$W_{wlc} = \frac{\Sigma^{-1} e^T}{e\Sigma^{-1} e^T} \tag{3.2}$$

$$\sigma_{wlc} = \frac{1}{e\Sigma^{-1} e^T} \tag{3.3}$$

where $\mathbf{e} = [1, 1, ..., 1]$ is a $1 \times N$ vector, $X = [x_1, ..., x_N]$, X_{wlc} is the fused estimation, σ_{wlc} is the variance of the fused estimation and Σ is the covariance matrix of the observations. This solution exists if $e^T \Sigma^{-1} e$ is non-singular or Σ is positive definite.

The output of a sensor can be also modeled as a *probability mass function* (pmf). Such sensor models are called probabilistic sensors. Probabilistic CP provides a common framework under which different probabilistic sensors can exchange data, and looks after methods of merging pdfs consistently. Linear and Logarithmic Opinion Pool (LOP and LGP) are two popular [54, 84, 80, 79, 29, 32, 90, 88] methods for probabilistic CP.

The difference between these two methods are in several issues. The point estimation methods are computationally less expensive than probabilistic methods while the later are more accurate. The probabilistic sensors provide more information comparing with point estimation. Therefore, the decisions based on a probabilistic sensor is more reliable than a point estimation method.

In the following sections, first we evaluate point estimation, then, we overview LOP and LGP and introduce two probabilistic CP methods.

3.4 Cooperative Perception

In the sequel, after defining some concepts, we overview two popular methods for probabilistic CP (Linear Opinion Pool (LOP) and Logarithmic Opinion Pool (LGP)).

3.4.1 Problem formulation

We assume that output of a sensor is in the form of a pdf. The pdf can represent a likelihood or an a posteriori function.

Consider a scenario where a group of N distributed sensors are measuring a quantity of interest. In general, the sensor measurements assumed to be uncertain. Sensor i provides a probability vector $\mathbf{P_i}$ size $M \times 1$ over the state space as its measurement model and M is the number of states. For a D-dimensional state space (D>1), we vectorize the state space matrix. The sensor observations are fused together using CP method f:

$$f : \mathbf{P_1}, \mathbf{P_2}, ..., \mathbf{P_N} \longmapsto \mathbf{P} \tag{3.4}$$

where $\mathbf{P}$ is also a pdf. The main difficulty is to find an appropriate function f. Actually, there can be an unlimited number of candidates for function f. It can be a simple analytical function, e.g., arithmetic or geometric mean. Note that the resulting pdf does not need to be similar to any of the fused PVs.

3.4.2 Agreement and Disagreement

There is agreement between the observations $\mathbf{P_1}$ and $\mathbf{P_2}$ of two sensors if:

$$d(\mathbf{P_1}, \mathbf{P_2}) \leq \xi \tag{3.5}$$

where $d(.)$ is a distance measure between the two observations and ξ is a positive number. The distance measure can be defined as Kullback-Leibler distance[49], Bhattacharrya distance[13] or other. Otherwise we say that there is disagreement among the two observations.

There are differences between outliers and disagreement. Outliers are the observations that are provided by a faulty sensor. This could be anything, including the reason for disagreement. However, there are situations where groups of sensors disagree. If the size and uncertainty of the groups are comparable, then we have a disagreement among the sensors. We can not ignore group observations without providing more evidences.

3.4.3 Observation Uncertainty

When we are not sure about a fact, we say that we are uncertain about it. Uncertainty in sensor measurement can be due to noise, imperfection and etc. In this case, the actual state is unknown and the sensor reading is uncertain. In order to reduce the uncertainty, observations of several sensors can be fused.

3.4.4 Measure of Observations Uncertainty

A major problem in CP is how to measure observations uncertainty. When the uncertainty has Gaussian form, the covariance matrix can represent the uncertainty. When the form of uncertainty is non-Gaussian, we use entropy [77] to measure the amount of uncertainty, e.g., for $\mathbf{P}_i$:

$$H(\mathbf{P}_i) = -\sum_{j}^{M} P_{ij} \ln P_{ij} \tag{3.6}$$

Entropy can be considered as a measure of uncertainty in a random event . For example it can show a measure of the randomness of a random variable or if a data source such as a probabilistic sensor expresses its measurement in form of a probabilistic distribution, the entropy of the distribution represents scatterdness of a probability distribution. The discrete entropy is an absolute measure of randomness. Entropy is a positive number. When entropy is zero we are fully certain about the outcome of L. As the value of H increases up to 1 we become less certain.

3.4.5 Entropy as Measure of Observation Quality

We assign a weight to each sensor observation. Weights express our certainty in observations and an agent who desires to fuse the information of different sensors should take them into consideration. If we quantify the quality of observations, we assign larger weights to better observations when fusing them. Entropy can be interpreted as the quantification of the uncertainty associated to an observation [1] [19] [97]. In CP, we use the concept of entropy to quantify the quality of observations. Lower entropy observations are better than higher entropy observations[1]. For N measurements $\mathbf{P_1}, \mathbf{P_2}, ..., \mathbf{P_N}$, a weight w_i is assigned to observation $\mathbf{P_i}$ by sensor i that reflects the quality of the observations.

$$w_i = \frac{1}{H(\mathbf{P_i})^k} \tag{3.7}$$

where $1 \leq i \leq N$ and $k > 0$. Larger k assign larger relative importance to more certain observations.

[1] Though lower entropy does not necessarily mean better accuracy for a sensor (e.g., if the sensor is biased) it is in general desirable, normally if one assumes, as it is done here, unbiased sensors.

3.4.6 Entropy as a Measure to Qualify the CP Algorithm

Entropy can also be used to evaluate the quality of a fusing algorithm. Some researchers used entropy as a measure to evaluate a CP algorithm [101][33].

If there is agreement among observations, the entropy of the fused observations is less than or equal to the minimum entropy of individual observations. In other words, for N observations $\mathbf{P_1}, \mathbf{P_2}, ..., \mathbf{P_N}$:

$$H(f(\mathbf{P_1}, \mathbf{P_2}, ..., \mathbf{P_N})) \leq min(H(\mathbf{P_1}), H(\mathbf{P_2}), ..., H(\mathbf{P_N})) \tag{3.8}$$

On the other hand, if the entropy of the fused observations is less than or equal to the minimum entropy of individual observations, there may be agreement or disagreement among observations. However, it is computationally costly if we first fuse the observations and then find and remove the faulty observation In practice, first, we have to find and remove the faulty observations. This can be done by grouping them. We measure the distance between the observations and the observations which have smaller relative distance are put in a group. We try to combine the groups which are close enough. At final steps we keep the groups and remove the individual observations. In Fig.3.1 three observations $\mathbf{P_1}, \mathbf{P_2}, \mathbf{P_3}$ are drawn. We use LGP (see, Subsection 3.4.8) to fuse the observations. Fusing $\mathbf{P_1}$ and $\mathbf{P_2}$ results in an observation with lower entropy but fusing $\mathbf{P_1}$ and $\mathbf{P_3}$ delivers an observation with higher entropy. Therefore $\mathbf{P_1}$ and $\mathbf{P_2}$ agree on the measurement, contrary to $\mathbf{P_1}$ and $\mathbf{P_3}$.

It is possible that the entropy of the fused observations is smaller than entropy of individual PVs while there is a disagreement among them.

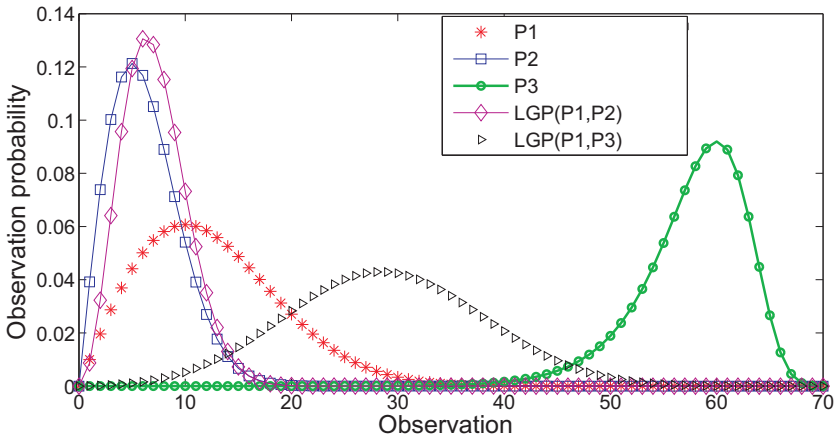


Figure 3.1: *Three observations* $\mathbf{P_1}, \mathbf{P_2}$ *and* $\mathbf{P_3}$ *for which only* $\mathbf{P_1}$ *and* $\mathbf{P_2}$ *agree. LGP*$(\mathbf{P_1}, \mathbf{P_2})$ *has smaller entropy comparing to* $\mathbf{P_1}$ *and* $\mathbf{P_2}$*. LGP*$(\mathbf{P_1}, \mathbf{P_3})$ *has larger entropy than* $\mathbf{P_1}$ *and* $\mathbf{P_3}$*. H*($\mathbf{P_1}$)*=4.67, H*($\mathbf{P_2}$)*=3.65, H*($\mathbf{P_3}$)*=4.27, H(LGP*($\mathbf{P_1}, \mathbf{P_2}$))*=3.59 and H(LGP*($\mathbf{P_1}, \mathbf{P_3}$))*=5.22*

3.4.7 Linear Opinion Pool

One way to merge different observations from many sources is to take a linear weighted sum of observations. This method is known as Linear Opinion Pool(LOP)[54] [84] [80]. LOP is easy to implement and can be used to combine the output of several sensors. For N observations $\mathbf{P_1}, \mathbf{P_2}, ..., \mathbf{P_N}$, LOP is defined as:

$$\mathbf{P} = \sum_{i=1}^{N} w_i \mathbf{P_i} \tag{3.9}$$

where $\mathbf{P}$ is a PV and w_is are weights $w_i \geq 0$ that add up to one, as in (3.7).

The Mean and Variance of LOP are given by [96]:

$$E[\mathbf{P}] = \sum_{i=1}^{N} w_i E[\mathbf{P_i}] \tag{3.10}$$

$$Cov[\mathbf{P}] = \sum_{i=1}^{N} w_i Var[\mathbf{P_i}] + \sum_{i=1}^{N} w_i (E[\mathbf{P_i}] - E[\mathbf{P}])(E[\mathbf{P_i}] - E[\mathbf{P}])^T \tag{3.11}$$

$E[\mathbf{P}]$ and $Var[\mathbf{P}]$ are the expectation, variance and covariance of the probabilistic vector $\mathbf{P}$ respectively and T stands for transpose. The expectation $E[\mathbf{P_i}]$ of a probability vector $\mathbf{P_i} = (P_{i1}, P_{i1}, ..., P_{i1})^T$ is defined as $E[\mathbf{P_i}] = (E[P_{i1}], E[P_{i2}], ..., E[P_{im}])^T$ and $Var[\mathbf{P_i}] = E((P_{i1} - E[P_{i1}], P_{i2} - E[P_{i2}], ..., P_{im} - E[P_{im}])^T (P_{i1} - E[P_{i1}], P_{i2} - E[P_{i2}], ..., P_{im} - E[P_{im}]))$ and T is the transpose operator. $E[P_{im}]$ is the expectation of the random variable P_{im}.

According to (3.10) and (3.11), the mean of LOP is the weighted sum observations mean and the variance of LOP is the sum of two terms. The first term can be interpreted as a weighted sum of observation variances or within-model variance, and the second term can be interpreted as between-model variance. In other words the second term expresses the deviation from the mean of LOP and is called disagreement term. It is clear from the above discussion that the variance of LOP is equal to or greater than the minimum variance. If we consider the variance as a measure of uncertainty, LOP can not decrease the uncertainty. Variance of LOP is always equal or greater than the minimum variance of observations and that is its main disadvantage. Another problem with LOP arises when one sensor delivers a wrong measurement, especially when we assign the same weight to all of the observations or a higher weight to the false observation due to its lower relative uncertainty.

3.4.8 Logarithmic Opinion Pool

Another approach to fuse sensor observations is *Logarithmic Opinion Pool*(LGP)[29] [32] [70]. If we assign a weight to each observation measuring its uncertainty then, for N observations, LGP is defined as:

$$\mathbf{P} = \frac{\mathbf{P_1}^{w_1} \circ \mathbf{P_2}^{w_2} \circ ... \circ \mathbf{P_N}^{w_N}}{(\mathbf{P_1}^{w_1} \circ \mathbf{P_2}^{w_2} \circ ... \circ \mathbf{P_N}^{w_N})\mathbf{e}} \tag{3.12}$$

where $\mathbf{e} = [1, 1, ..., 1]^T$ is a $1 \times m$ vector, w_is are weights as in (3.7), and $\circ$ is Hadamard product[2] of two matrices. $\mathbf{P_i}^{w_j}$ is a vector element wise power operation.

Comparing to LOP, LGP is less scattered and more certain. Another preference of LGP over LOP is that merging the observation of N sensors using LGP is the same as fusing $N-1$ measurements first and then fusing the result with the N^{th} measurement. A major problem with LGP is that if one of the sensors assigns zero probability to a point in the state space, no matter how many sensors contribute and also their vote, LGP for that point is zero. This is undesirable when the output of one of the sensors becomes close to zero due to failure or unpredictable error. Another drawback of LGP is the assumption of independence of the individual pdfs. It is a difficult assumption to satisfy and only true when each sensor measures different features.

3.5 Proposed Fusion Methods

In this section we introduce two new methods to overcome the difficulties of LOP and LGP: Cooperative Opinion Pool (COP) and p-norm Opinion Pool (POP). The context of our work is Networked Robot Systems (NRS), examples of which are a team of mobile robots, or a network of "smart" surveillance cameras. In this environment, a number of agents cooperatively and collaboratively perform a set of allocated tasks. They are able to communicate but their communication range is limited. The fusion algorithm should be designed to cope with the sensor fusion problem in such an environment. An important issue in information fusion in such environment is sensor disagreement and failure [45, 2]. This may affect the

[2]For two matrices A and B, Hadamard product of two matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ of the same size is just their element-wise product $A \circ B \equiv [a_{ij}b_{ij}]$.

decision of a subteam and consequently the agent team. The CP method should be able to recognize and filter out these faulty measurements and also resolve disagreement among the information sensors.

Using LOP and LGP methods in a NRS might be problematic, e.g., because they do not distinguish between situations of agreement vs. disagreement between sensors. We evaluate the compliance of these two fusion methods with our specifications. To overcome their disadvantages, we introduce a probabilistic sensor fusion algorithms for fusion of information which generalizes LOP and LGP.

Both of the methods, COP and POP, consider all the dependencies among observations, such as LOP, and reduce the uncertainty, such as LGP. However, COP needs more computational power compared to POP. On the other hand, COP is more flexible compared to POP. POP is designed to meet some properties. We check and compare the performance of the CP methods by running different simulations. Here, we assume a probabilistic model for sensors. In other words, sensor delivers its measurement in the form of a pdf. Moreover, all observations are assumed to refer to a common global frame. We also assume that the majority of sensors is unbiased. This is a very weak assumption comparing to the usual assumption that all sensors are unbiased and independent, e.g., as in [7].

3.5.1 Cooperative Opinion Pool

To overcome the difficulties of LOP and LGP we introduce a new approach which we designate as Cooperative Opinion Pool (COP).

Let $\mathbf{P_1}, \mathbf{P_2}, ..., \mathbf{P_N}$ be a set of PV related to observations of N sensors $S_1, S_2, ..., S_N$ respec-

tively. The COP of those observations is defined as:

$$\begin{aligned}\mathbf{P} = K((\sum_{i} w_i \mathbf{P_i})^{\frac{1}{\alpha_1}} + (\sum_{i,j,i\neq j} (w_i + w_j)\mathbf{P_i} \circ \mathbf{P_j})^{\frac{1}{\alpha_2}} \\ +(\sum_{i,j,k,i\neq j\neq k} (w_i + w_j + w_k)\mathbf{P_i} \circ \mathbf{P_j} \circ \mathbf{P_k})^{\frac{1}{\alpha_3}} + ... \\ +(\mathbf{P_1} \circ \mathbf{P_2} \circ ... \circ \mathbf{P_N})^{\frac{1}{\alpha_N}})\end{aligned} \tag{3.13}$$

where $1 \leq i, j, ..., k \leq N$, $\sum_{i=1}^{N} \alpha_i = 1, 0 \leq \alpha_i \leq 1, w_i > 0$, as in (3.7), and K is a normalization factor that is determined by the requirement that COP is a PV. Once again, w_is weight the measurement uncertainty of sensor i, while α_is are tuned as explained below.

Let us call each term in COP a component $C_i, 1 \leq i \leq N$. The purpose of the formula for COP is to weight all possible combinations of the team sensors to avoid LOP and LGP problems. Each component C_i includes $\frac{N!}{i!(N-i)!}$ terms and computes redundant information among i sensors. Accordingly C_N represents N sensors and intuitively is the most valuable and important component of COP. For this reason one should usually assign to α_N the largest value among the α_is. Among the other terms, C_{N-1} is the most important component. If C_N is zero or very small, it means that at least one sensor disagrees with the others. In this case C_{N-1} presents the most important term. C_1 only becomes important when there is general disagreement between sensors. In such situation in COP, all the components become zero but C_1. The remaining components correspond to some of the sensors sharing the same part of observations. In a sense these terms explain the agreement of q sensors, with $1 < q < N$.

Let us consider an example of fusing the observations of 5 distance sensors in agreement. The measurements obtained from the sensors while observing the same object are shown in Fig. 3.2. Without lack of generality the outputs of the sensors are considered 1D Gaussian. They have different means and standard deviations due to noise, sensors quality, calibra-

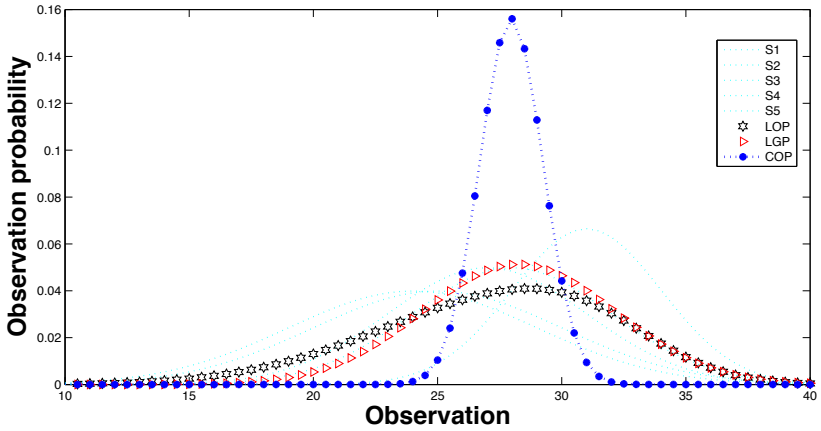


Figure 3.2: *An example of fusing 5 observations. Observations are 1D normally distributed but have different means and standard deviations* ($\mu_1 = 25, \sigma_1 = 5, \mu_2 = 27, \sigma_2 = 4, \mu_3 = 24, \sigma_3 = 5, \mu_4 = 28, \sigma_4 = 4, \mu_5 = 31, \sigma_5 = 3$). *LOP, LGP and COP (k=2,* $\alpha_n/\alpha_{n-1} = 2$) *are plotted.*

tion, etc. The result of fusing the observation of those sensors using LOP, LGP and COP is also shown in the figure. Comparing to the other two algorithms, COP has the smallest entropy.

Another comparison of three CP methods is shown in Fig. 3.3. In this example a robot which is equipped with three distance sensors in agreement, moves around and at each time instant (step) measures distance to different objects in the environment. Outputs of the three sensors are assumed 1D Gaussian with different means and variances (not only between sensors but also between observations for the same sensor over time, as the distance to an object will change, with an impact on mean and variance). To improve the uncertainty of the observations we fuse the information from the three sensors at each observation location using LOP, LGP and COP. We consider the ratio of (entropy of fused observations)/(entropy of the best individual sensor observation) as a performance factor.

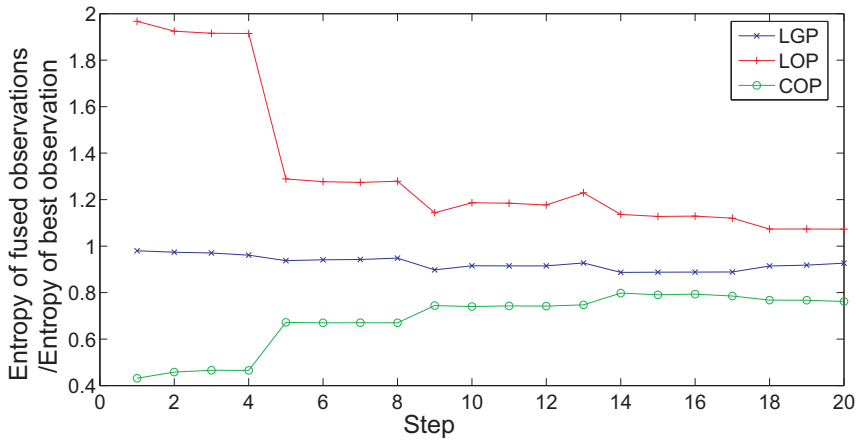


Figure 3.3: *A comparison of performance of three CP methods: LOP, LGP and COP. Entropy of LOP is always larger than the entropy of the best individual observation. Setting α_i and w_i to appropriate values, the performance of COP can be better than LGP. In this example COP always delivers an observation with less entropy than the best individual observation*

We evaluate the methods based on this factor. As seen from the figure, entropy of LOP is always larger than the entropy of the best observation while entropy of LGP is always equal to or smaller than the entropy of the best observation. Entropy of COP is always equal to or smaller than the entropy of the best observation.

3.5.1.1 Handling Disagreement

In Fig. 3.2 LOP, LGP and COP of 5 sensors agree. In this situation, the pdfs of all 5 observations are drawn. If the estimated distance to the real position of the object is considered at the maximum of the pdf, then these three methods deliver approximately the same estimates. The PV obtained by COP is sharper than the others meaning that it is more certain about the position of the object. What would happen using COP if one sensor makes a

mistake and disagrees with the others?

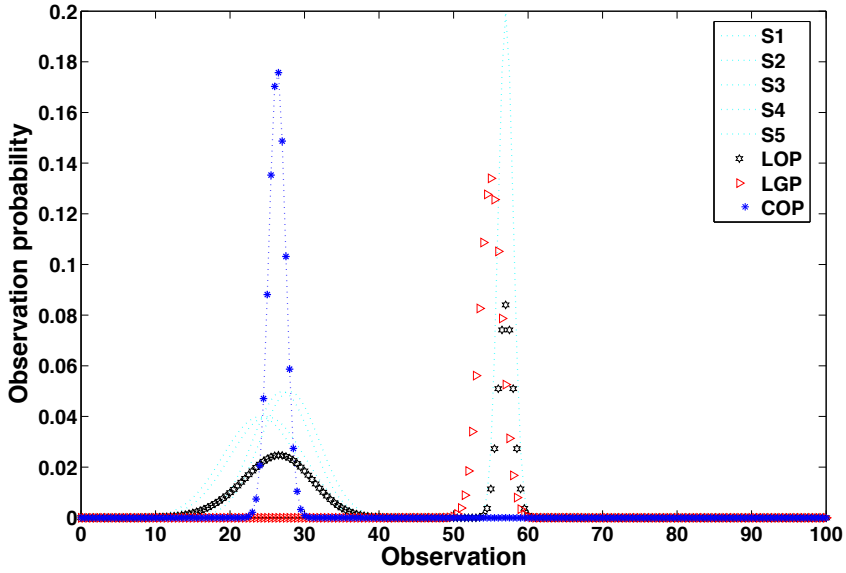


Figure 3.4: *An example of fusing 5 observations* $(S_1 - S_5)$. *Observations are 1D normally distributed but have different means and standard deviations* ($\mu_1 = 25, \sigma_1 = 5, \mu_2 = 27, \sigma_2 = 4, \mu_3 = 26 \sigma_3 = 5, \mu_4 = 57, \sigma_4 = 1.5, \mu_5 = 59, \sigma_5 = 2$). *LOP, LGP and COP (k=2,* $\alpha_n/\alpha_{n-1} = 2$*) are plotted. Note that LOP produces a bimodal pdf, with a global maximum between 50 and 60 distance units.*

Fig. 3.4 shows the observations and also the results of the three CP rules. Logically when four $(S_1 - S_4)$ out of five observations are close and S_5 has no intersection or only intersects in a small region with the rest, the 5th observation is considered as a failure. In this situation LGP and LOP fail and they provide a completely wrong final pdf. Fig. 3.5 shows another example where LOP and LGP fail. In this example, the output of three observations are close but two others disagree. COP considers intersection of the three observations as the most probable region for position of the object, although it considers a smaller proba-

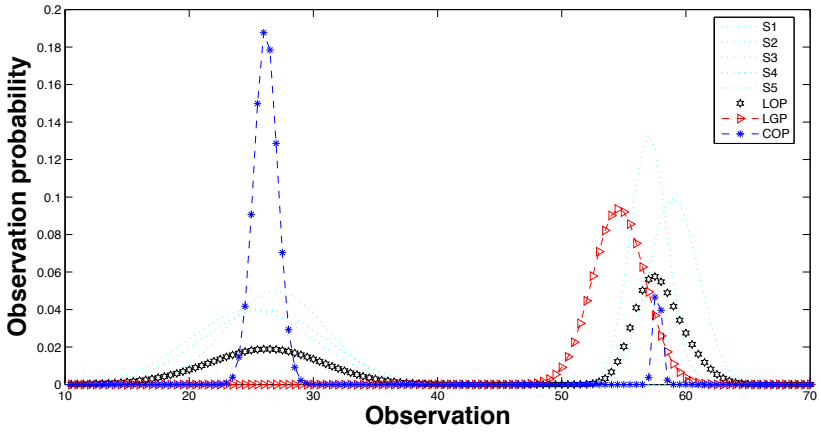


Figure 3.5: *An example of fusing five observations. Observations are normally distributed but have different means and standard deviations ($\mu_1 = 25, \sigma_1 = 5, \mu_2 = 27, \sigma_2 = 4, \mu_3 = 24, \sigma_3 = 5, \mu_4 = 44, \sigma_4 = 4, \mu_5 = 47, \sigma_5 = 1.5$). LOP, LGP and COP (k=2, $\alpha_n/\alpha_{n-1} = 2$) are drawn.*

bility for the intersected region of two other observations but total uncertainty of the COP is smaller than the best individual observation.

3.5.1.2 Computation Cost

Among the three methods, LOP and LGP are computationally cheaper than COP. Computational complexity of LOP and LGP is $O(M(2N-1))$. Computational complexity of COP is $O(MN2^N)$. M is the size of PVs domain and N is the number of observations.

However, in practice, COP can be often computed by considering only the first two non-zero components. For example, in the case of agreement between N observations and taking $\alpha_{i+1} \gg \alpha_i$ $(e.g.,\ \alpha_{i+1} = 2\alpha_i)$, we only need to calculate C_N and C_{N-1} and the rest becomes negligible. In the case of a disagreement between one of the sensors and the rest, we only need to calculate C_{N-1} and C_{N-2} because C_N becomes zero due to the disagree-

ment. Moreover in a team of N sensors, CP of N observations is frequently not necessary. By dividing a team to sub-teams, CP of a small number of observations is enough.

3.5.1.3 Setting parameters

As we explained earlier, in COP we set $\alpha_i = \frac{\alpha_{i-1}}{\gamma}, 0 \leq \gamma \leq 1$ $(e.g.,\ \alpha_i = 2\alpha_{i-1})$. Therefore we only need to set one more parameter comparing to LOP and LGP, i.e., the weights or, indirectly k in equation (3.7). Its value depends on the problem. If the probability of failing sensors is high and they may deliver a wrong observation, take $0 \leq k \leq 1$. If the sensors are reliable we take $k \geq 1$.

3.5.1.4 Simulation Results

To evaluate the performance of COP we ran a simulation of a number of robots working as a team. They are able to communicate to each other and can share information via a blackboard. The world was divided into equal squares. We considered different resolutions ranging from $50 * 50$ to $200 * 200$ units. Robots equipped with two types of sensors (vision sensor and odometry sensors) were simulated. The vision sensor measurement model is a 2D Gaussian. The mean is centered on the real position of the object and the standard deviation is considered a function of distance (units in cells):

$$\sigma_{xx} = \sigma_{yy} = \begin{cases} 8|d - 7.5| & 3 \leq d \leq 11 \\ 20 \lg |d - 7.5| & 1 < d < 3 \ or \ 11 < d < 13 \end{cases} \tag{3.14}$$

and $\sigma_{xy} = \sigma_{yx} = 0$.

The range of the sensor is restricted to cells 1 to 13 ahead of the robot sensor. Sensor delivers the best observation between 3 and 11, where the entropy is lowest. By increasing or decreasing distance, the uncertainty of observation increases. We also considered

a 2D Gaussian model for odometry sensor. The mean is centered on the real position of robot. The standard deviation increases in the direction of motion of the robot. To prevent excessive error in the long run, we re-initialized the odometry sensor when the travelled distances exceeded 5 cells. At each step, robots in each sub-team first update their own belief using sensory information and then by considering odometry data. Then, robots exchange information with teammates. Using this strategy we not only reduced the entropy of localization of a common object but also in the situations where robots were not able to observe the object, got a belief via neighbors. Besides that, we needed less computational power, memory and communication than any Markov localization method for Multi-Robot Localization. Robots are equipped with two types of sensors: vision and odometry. Sensors have a limited range and an uncertainty is associated with each observation that depends on the relative position of the robot and object. We studied the effect of cooperation of robots when a robot intends to reduce the uncertainty of observation for self-localization or common object localization and also if, for any reason, it is not able to locate the common object.

In this simulation the goal is to use information gathered by teammates to reach a consensus on the location of the robots and objects in the field of view of the team. However improving object localization is another benefit of this simulation. For those robots that have a more uncertain view of an object comparing to other teammates, there is a chance to fuse their uncertain data with other teammates in order to refine it.

In this simulation, robots use a multi-robot version of Markov Localization Algorithm (MLA) to self-localize and to localize objects on the field [70][59]. The pseudocode of the algorithm which is described in Table 3.1 is adapted from [59]. Each robot is confronted with two types of uncertainties: uncertainty in self-localization and uncertainty in

object localization. The main goal of this simulation is to reduce these two types of uncertainties. In a team of N robots, $Bel_n^{n,t}(L^t = l) = P(L_n^t = l|d_n^t)$ denotes the belief of the n^{th} robot about its own location at time t being l and receiving information d_n^t, e.g., the information received from the odometry sensor. If robot n receives information about the location of the object Obj from a second robot m with the goal of reducing its uncertainty, it must take into account the observation of robot m and fuse the two observations in order to reduce the uncertainty. The belief which is obtained from CP of the two beliefs is called cooperative belief. It is defined as:

$$CBel_{Obj}^{n\Leftarrow m}(l) = P(L_{Obj} = l|o_n, o_m) \tag{3.15}$$

where o_n and o_m are observations of n^{th} and m^{th} robots respectively. Based on this we can define cooperative entropy:

$$H(CBel_{Obj}^{n\Leftarrow m}) = -\sum_l CBel_{Obj}^{n\Leftarrow m}(l) ln CBel_{Obj}^{n\Leftarrow m}(l) \tag{3.16}$$

Here we use COP definition (3.13) to calculate the cooperative belief, while in [59], a different method was used. Entropy of the cooperative belief is used to measure the amount of uncertainty of the belief. Naturally we expect a reduction in entropy of cooperative belief comparing to the individual beliefs if there is agreement among the observations.

Although a full cooperation among all the robots of a team can improve individual robot estimates of object locations, it is computationally expensive and sometimes technically impossible. Other reasons to divide a team into groups of subteams are that communication band limitations, communication range restrictions and also energy saving. To exchange information between all team members, a wider band is needed. If the team is deployed in a large area, a direct contact between every two teammates may not be possible, due to range

Do ***forever***

Do for each robot

register robot in sub-teams

End Do

Do for each sub-team

Self localize robots using Markov Localization algorithm [70], obtaining $Bel_n^{n,t}(L^t = l|o_n)$.

Send information to sub-team blackboard.

Update beliefs for each robot in sub-team, based on other teammate observations, using COP.

Exchange information among sub-teams.

Update beliefs for each robot in sub-team, based on other sub-teams observations, using COP.

end Do

Do for each sub-team

Localize Objects in the field of view using observation model.

Send information to sub-team blackboard.

Update beliefs about the objects in the field of view using COP.

Exchange information among sub-teams.

Update beliefs for each robot in sub-team, based on other sub-teams observations, using COP.

end Do

end forever :

Table 3.1: *Cooperative Localization Algorithm pseudocode*

limitations and/or occlusions. For this reason we divide a team into sub-teams. In each sub-team, a robot acts as the subteam leader. It receives data from the teammates, fuses and distributes data among members. It has also the duty of the exchanging data with other subteams. In the cases where the reliability is a matter of concern and the possibility of losing a subteam leader is high, we can consider a redundant leader for the subteam. This way a subteam has also a backup and in the case of loosing the main leader the subteam leader can take the responsibility. Besides that, different sub-teams may have common members. Taking advantage of this, two sub-teams can exchange data through a common member.

The conflict between observations can be resolved by exchanging the data with other sub-teams. In the algorithm, first each robot registers itself in sub-teams. Then each robot self-localizes using its own observations, send sub-team information to the sub-team blackboard and fuse sub-team observations. To fuse observations we use the COP algorithm. After this stage, robots in a sub-team have the same belief about the localization of their teammates. Next, we exchange information of sub-teams and fuse them using COP. We repeat the process for the objects in the field of view of the team. At this point, the full team has the same belief about teammate location and object localizations.

The world was divided into a grid of equal squares. We considered different resolutions ranging from 50×50 to 200×200 squares. The vision sensor measurement model is a 2D Gaussian. The mean is centered on the real position of object and the standard deviation is considered a function of distance (units in cells):

$$\sigma_{xx} = \sigma_{yy} = \begin{cases} k_1|d - k_2| & a_2 \leq d \leq a_3 \\ k_3 log|d - k_4| & a_1 \leq d < a_2 \; or \; a_3 < d \leq a_4 \end{cases} \tag{3.17}$$

and $\sigma_{xy} = \sigma_{yx} = 0$.

Thus, the covariance matrix is:

$$\begin{bmatrix} \sigma_{xx}^2 & 0 \\ 0 & \sigma_{yy}^2 \end{bmatrix}$$

The range of the sensor is restricted to cells a_1 to a_4 ahead of the robot sensor. The

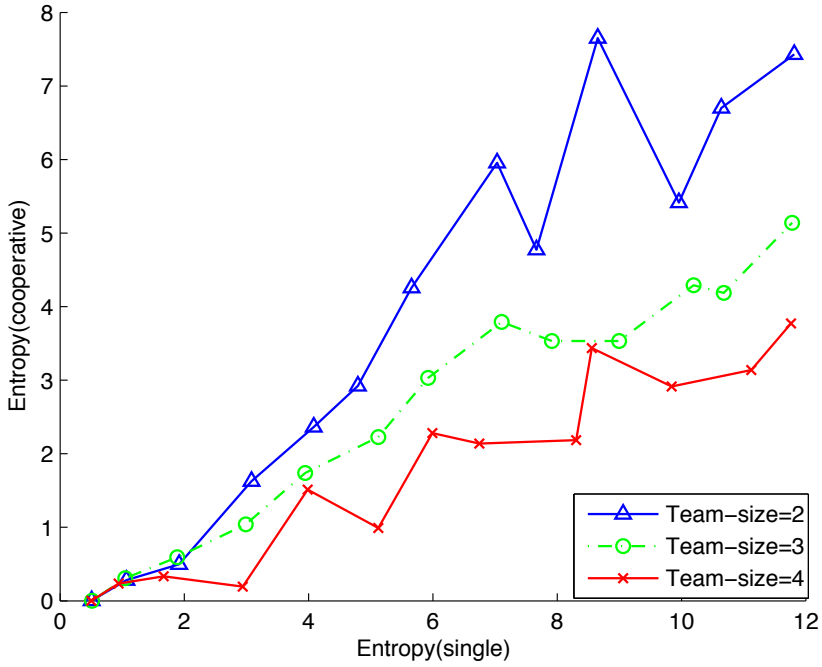


Figure 3.6: *Entropy of multi-robots localization team versus single robot localization using COP. Team size ranged from 2 to 4.*

sensor delivers the best observation between a_2 and a_3, where the entropy is the lowest. By increasing or decreasing distance beyond this interval, the uncertainty of observation increases. We also considered a 2D Gaussian model for the odometry sensor. The mean is centered on the real position of robot. The standard deviation increases in the direction of

motion of the robot. To prevent excessive error in the long run, we reinitialize the odometry

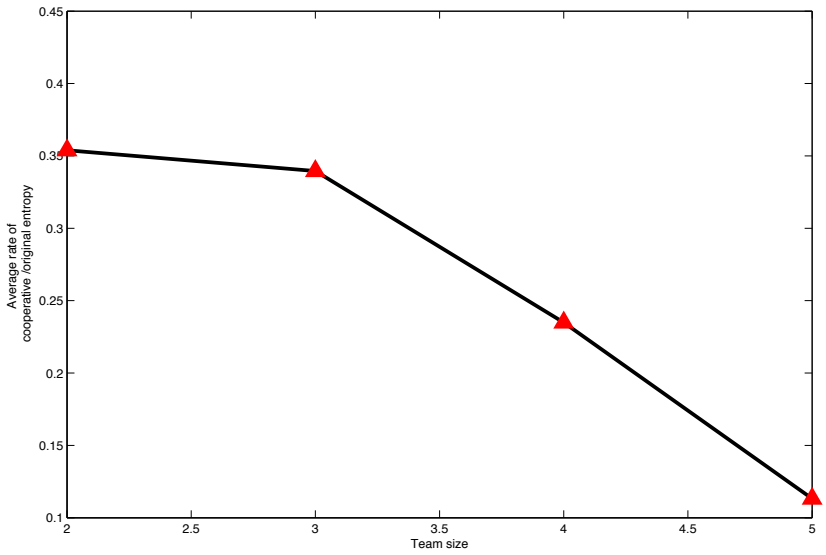


Figure 3.7: *Average entropy of localization versus size of team using COP.*

sensor when the traveled distances exceeded 5 cells. In the experiment, we studied the effect of cooperation on mutual localization and on the localization of a common object. In Fig. 3.6, the plot of entropy of the single robot belief versus the entropy of the cooperative belief is shown for $a_1 = 1, a_2 = 3, a_3 = 11, a_4 = 13, k_1 = 8, k_2 = 7.5, k_3 = 20$ and $k_4 = 7.5$. As the number of contributors increases we see a reduction in the amount of entropy. In Fig. 3.7, we can see the average entropy ratio (cooperative entropy/original entropy) decreases as the number of cooperative robots increases.

3.5.2 P-Norm Opinion Pool

In this subsection, we introduce an axiomatic CP method to combine the measurements received from several probabilistic sensors, taking into account that some of them might fail[2]. Fusion of such information can lead to disagreement among sensors and generates large error. Here, first, we define the basic requirements of a CP method. By that we mean the properties that the method should have in order to produce the desirable results. In particular, we use $p-norm$ (also called $lp-norm$) to fuse the information. We call this method *p-norm opinion pool* (POP) and check its compliance with the defined specifications. This method successfully overcomes the shortcomings of LOP and LGP fusion methods, as demonstrated in simulated as well as real-world experiments. Then, we propose and formalize the POP method. We check and compare POP compliance with COP, LOP and LGP. We also introduce a method to automatically adjust and set the POP parameters. We check the POP method in presence of sensor failure through the simulation.

3.5.2.1 Motivation example

The following example shows how some CP methods, e.g. using Weighted Linear Combination of observation (WLC)[7] and standard Bayesian methods are sensitive to sensor failure. In this example, we attempt to track a robot by the information provided by the robot odometry sensor and a number of ceiling mounted cameras. All information is routed to a central processing unit. Sensors noise is considered to be Gaussian. To estimate the robot position, first, the information received from the cameras is fused. Then, by using a filter, the fused information and the odometry, the position of the robot is estimated. In Fig.3.8(a), the real path and two estimated paths are shown. The first estimate is obtained

when WLC is used to fuse the information and standard Kalman filter is used to determine the final estimate. In the second case, a standard Bayesian method is used to fuse the information and MCL is used to estimate the path. In this simulation, we assume that the sensors are unbiased. As we can see from the figure, the estimated path is a good approximation of the actual path. Now, to see the effect of sensor failure, 30% of the cameras observing the robot are biased. Fig.3.8(b) shows the real and estimated positions. Similar examples are ran 100 times. The estimation error is defined as the Euclidean distance between the real position and the estimated positions. Comparing to the previous case, if we use WLC and the Kalman filter, we get a 10 times increase in estimation error and if we use the standard Bayesian fusion method and MCL, we have 17 times increase in error w.r.t. the real path . This proves that excluding the erroneous measurements and resolving the disagreement is a crucial issue.

3.5.2.2 The fusion specifications

Now, we define a set of specifications that a fusion algorithm should comply with. These specifications play the role of constraints that limit the number of potential candidates for f in (3.4). These specifications are considered for a NRS.

Based on these specifications, we evaluate the fusion method. In general, it might not be possible to find a function that fully satisfies all of them. So, we should find a fusion method that has maximum compatibility with the specifications.

1-**Uncertainty reduction**: A fusion rule should be defined in such a way that the uncertainty of fused observation is less than minimum uncertainty of the original observations if

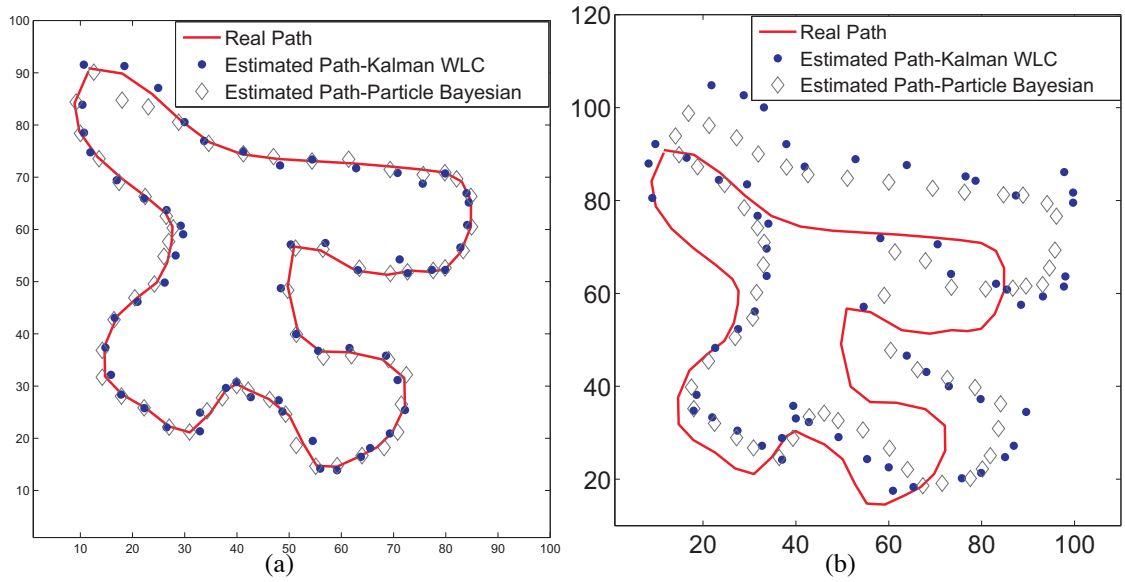


Figure 3.8: *Results of the motivation example for POP. A tracking scenario is considered in which a robot is tracked by a number of fixed cameras. The left plot is the result of tracking using two well known methods when sensors are unbiased. The right plot is the result of tracking when 30% of the sensors are biased.*

there is a general agreement among observations. For two observations $\mathbf{P}_1, \mathbf{P}_2$:

$$d(\mathbf{P_1}, \mathbf{P_2}) \leq \xi \Rightarrow \Psi(f(\mathbf{P_1}, \mathbf{P_2})) < min(\Psi(\mathbf{P_1}), \Psi(\mathbf{P_2})) \tag{3.18}$$

where Ψ is a measure for the uncertainty, e.g. the entropy and *min* stands for minimum. If there is disagreement among the measurements, depending on the number of disagreements and amount of disagreement, uncertainty of fused observation can be larger than the uncertainty of the best observation. Fusing inconsistent observations creates more hypotheses, which cause an increase in uncertainty of the fused observation.

2-**Fault tolerance**: One important issue in sensor fusion is detecting and excluding faulty measurements. Faulty measurements are defined as measurements whose distance from

actual value is more than a given threshold. To come up with this difficulty, we assume the majority of the sensors are unbiased.

3-**Zero preservation**: If probability of the fused measurement for a state is zero then all sensors should assign zero probability to the state.

$$\mathbf{P}_j = f(P_{1j},...,P_{Nj}) = 0 \quad \Rightarrow \quad \forall i \;\; P_{ij} = 0 \tag{3.19}$$

where $\mathbf{P}_{ij}$ represents the probability that sensor i assigns to the state j and $\mathbf{P}_j$ is the probability of fused observation for state j.

4-**Non-zero preservation**: If the probability of fused observation for a state is non-zero then then at least one of the measurements should assign a non-zero probability to the state.

$$\mathbf{P}_j = f(P_{1j},...,P_{Nj}) \neq 0 \quad \Rightarrow \quad \exists i \;\; P_{ij} \neq 0 \tag{3.20}$$

Where $\mathbf{P}_j$ is the probability of fused observation for state j. By providing an example, we show the importance of this specification. Consider N sensors are measuring a quantity of interest. Suppose all of them except one (which assigns a zero probability for a particular state) report a non-zero probability for that state. Logically, the fused observation should not assign a zero probability to the state, specially if N is large. This property is very important, especially when some sensors are faulty but we can not tell which ones. Unfortunately, some fusion approaches, e.g., Bayesian-like approaches do not comply with this specification.

5-**Sequential updating**: a fusion specification is called sequentially updating if the result of fusing N sensor measurements is the same as fusing $N-1$ sensor measurements with an N^{th} sensor measurement.

$$f(\mathbf{P}_1,...,\mathbf{P}_{N-1},\mathbf{P}_N) = f(f(\mathbf{P}_1,...,\mathbf{P}_{N-1}),\mathbf{P}_N) \tag{3.21}$$

6-**Monotonicity**: Let $\mathbf{P} = f(\mathbf{P}_1, ..., \mathbf{P}_i, ..., \mathbf{P}_N)$ and $\mathbf{P}' = f(\mathbf{P}_1, ..., \mathbf{P}'_i, ..., \mathbf{P}_N)$. f is monotone if $P_{ij} < P'_{ij}$ then $P_j < P'_j$. In other words, this property says any change in one of the measurements should directly and proportionally affect the fused observations.

3.5.2.3 LOP and LGP compliance

LOP complies with monotonicity, sequential updating, zero preservation and non-zero preservation specifications. Despite that, the uncertainty reduction and fault tolerance are not satisfied. LGP complies with monotonicity, sequential updating, zero preservation and uncertainty reduction specifications. However, it does not satisfy non-zero preservation and fault tolerance.

In subsection 3.5.2.2 a set of specifications is defined to ensure the fusion function possesses certain properties and found out that LOP and LGP do not satisfy some of the specifications. Besides these, in a NRS, the fusion specification should be able to cope with different situations. For example, in situations where there is a general disagreement among the sensors, LOP is preferred over LGP (because LGP does not comply with non-zero preservation). There are situations where there is agreement among the sensors and LGP is preferred (because LOP does not comply with uncertainty reduction). However, in some situations using only one of them may not be the solution. To overcome the difficulties of LOP and LGP and also provide a more generic and flexible probabilistic fusion method appropriate for a NRS, another probabilistic fusion method is introduced.

Here, first, we formalize the fusion problem as an optimization problem. After checking the compliance with the defined specifications, the solution can be used as a fusion method. As we mentioned before, the output of the sensors is considered as a pdf. Let $\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N$

be a set of PV related to observations of N sensors $S_1, S_2, ..., S_N$ respectively. Let $\mathbf{P}$ represents the fused pdf. Let $d(.)$ represents the distance between two pdfs. We define the following objective function:

$$f = \sum_{i=1}^{N} w_i d(\mathbf{P}, \mathbf{P}_i) \tag{3.22}$$

where $d(\mathbf{P}, \mathbf{P}_i)$ is the distance between the fused measurement and i^{th} sensor measurement and $0 \leq w_i \leq 1$ and $\sum_{i=1}^{N} w_i = 1$. w_i is a weight assigned to $d(\mathbf{P}, \mathbf{P}_i)$ to represents the importance of the i^{th} sensor measurement over other measurements. Actually, f is a function that measures average weighted distance between the original observation and the fused one. If f is minimized, $\mathbf{P}$ will become the closest pdf to all the measurements:

$$\mathbf{P} = \underset{\mathbf{P}}{argmin}\ f = \underset{\mathbf{P}}{argmin} \sum_{i=1}^{N} w_i d(\mathbf{P}, \mathbf{P}_i) \tag{3.23}$$

Now, the problem is to consider an appropriate distance function. It can be defined as:

$$d(\mathbf{P}, \mathbf{P}_i) = (\mathbf{P}^{\kappa} - \mathbf{P}_i^{\kappa})(\mathbf{P}^{\kappa} - \mathbf{P}_i^{\kappa})^T \tag{3.24}$$

and $-\infty < \kappa < +\infty$. [3] Here κ plays an important role. It allows us to include the different fusion rules under a similar framework. Replacing (3.24) in (3.23) and solving the minimization problem results in:

$$f = \sum_{i=1}^{N} w_i (\mathbf{P}^{\kappa} - \mathbf{P}_i^{\kappa})(\mathbf{P}^{\kappa} - \mathbf{P}_i^{\kappa})^T$$

$$\frac{\partial f}{\partial \mathbf{P}} = \sum_{i=1}^{N} 2 w_i \kappa \mathbf{P}^{\kappa - 1} (\mathbf{P}^{\kappa} - \mathbf{P}_i^{\kappa})^T$$

$$, \quad \frac{\partial f}{\partial \mathbf{P}} = 0 = \sum_{i=1}^{N} w_i (\mathbf{P}^{\kappa} - \mathbf{P}_i^{\kappa}) \Rightarrow \ \mathbf{P}^{\kappa} = \sum_{i=1}^{N} w_i (\mathbf{P}_i^{\kappa})$$

[3] A^b raises each element of A, A(i) to the power b (vector element-wise power operator).

$$\mathbf{P} = POP(\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N) = \eta(\sum_{i=1}^{N}(w_i \mathbf{P}_i^{\kappa}))^{\kappa^{-1}} \tag{3.25}$$

The resulting method (3.25) is designated as P-norm Opinion Pool (POP).

In (3.25), η is a normalization factor that is determined by the requirement that $\mathbf{P}$ is a PV.

Once again, w_i weights the measurement uncertainty of sensor i.

LGP and LOP are particular cases of POP.

In fact, if we make $\kappa \to 0$, we obtain *LGP*.

$$lim_{\kappa \to 0} log(\mathbf{P}) = lim_{\kappa \to 0} \frac{log(\sum_{i=1}^{N} w_i \mathbf{P}_i^{\kappa})}{\kappa} \tag{3.26}$$

After applying l'Hopital's rule:

$$\begin{aligned} lim_{\kappa \to 0} log(\mathbf{P}) = \; & lim_{\kappa \to 0} \frac{\sum_{i=1}^{N} w_i \mathbf{P}_i^{\kappa} log(\mathbf{P}_i)}{\sum_{i=1}^{N} w_i \mathbf{P}_i^{\kappa}} = \sum_{i=1}^{N} log(\mathbf{P}_i^{w_i}) \\ & = log \prod_{i=1}^{N} \mathbf{P}_i^{w_i} \quad \Longrightarrow \quad \mathbf{P} = \prod_{i=1}^{N} \mathbf{P}_i^{w_i} \end{aligned} \tag{3.27}$$

while if we make $\kappa = 1$, we get LOP.

Method / Spec.	*LOP*	*LGP*	*COP*	POP
Uncertainty reduction	NO	YES	YES	YES
Fault tolerance	NO	NO	YES	YES
Zero preservation	YES	YES	YES	YES
Non−zero preservation	YES	NO	YES	YES
Sequential updating	YES	YES	YES	YES
Monotonicity	YES	YES	NO	YES

Table 3.2: *Comparing probabilistic fusion method properties*

POP satisfies monotonicity, sequential updating, zero-preservation, non-zero preservation, uncertainty reduction and fault tolerance. In Table (3.2), the compliance of LOP, LGP and POP are specified.

3.5.3 POP and the Motivation Example

Here, we consider the robot localization example (in subsection **??**) and evaluate POP using the example. We use the same configuration (same sensor and error models, failure rates) except that we use a MCL to track the robot. We use POP to fuse the observations. The result of simulation is shown in Figure 3.9(a). The estimate is considered to be the weighted average particle. This example shows that comparing to a standard method such as WLC which considers covariance-based average, POP provides a better and less erroneous estimate. When the POP is used, we only observed 26% (fixed κ) and 8% (auto-adjusting κ which is explained in 3.5.5) increase in estimation error comparing to the case (Fig. 3.8(a)) when the sensors are unbiased and the Kalman filter is used to track the robot.

3.5.4 Computation Cost

Here, the computational cost of calculating the weights is not considered because it depends on the way we assign weight to observations. Moreover, the weight allocation cost is the same for all three methods. Among the three methods, LOP is computationally the cheapest and POP is the most expensive one. However, the difference is not meaningful, especially if the number of states is large enough. If N and M represent the number of observations and the size of PV respectively, the computational cost of LOP is $O(M(2N-1))$. The

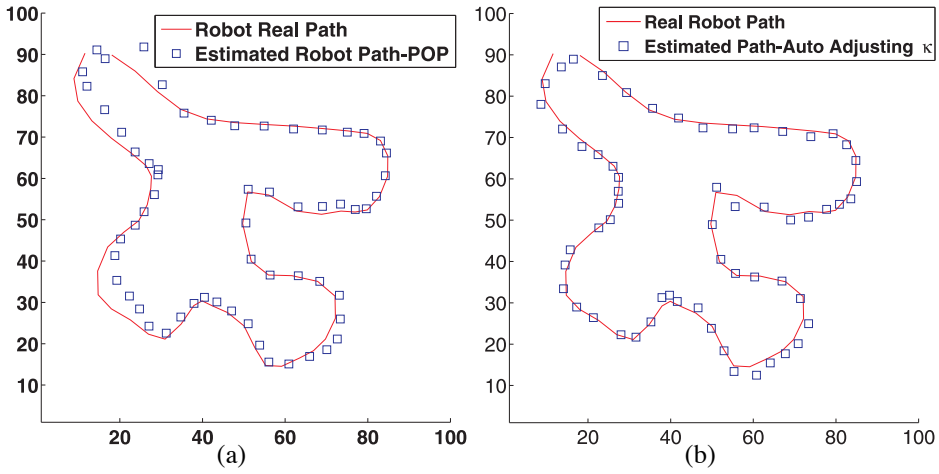


Figure 3.9: *The figures represent the results of the tracking example but this time we use POP and the MCL. The left figure shows the result when we use a fixed* κ *(*$\kappa = .1$*). In the right figure* κ *is dynamically set using the auto-adjusting algorithm Table3.3. Compared to Fig.3.8(b), a noticable reduction in estimation error can be seen.*

computational cost of LGP and POP is $O(M(2N+1)-1)$ and $O(2M(N+1))$ respectively. It is clear when $N >> 1$, the three methods impose the same cost. When N is small, there is a difference, although it is negligible.

3.5.5 Mechanism of determining κ

An important issue is how to choose an appropriate value for κ. As we explained earlier, different κ generates different results. For example, by taking κ close to one, the result is similar to LOP method. Taking κ close to zero, generates results similar to LGP. Therefore, depending on the problem, we have to take different values for κ. However, using a constant κ may not be a good option in a NRS environment. In such environment, we may have different situations over time. Therefore, we need a mechanism to set κ automatically.

Algorithm set $\kappa(O,\alpha,\beta)$

A: Group The Observations

To each observation $o_k \in O$ assign an index i and let $I = \{i\},\ 1 \le i \le N$ be the set of indexes

Calculate $D = \{d(o_j,o_k)\},\ 1 \le j,k \le N,\ j \ne k$ where $d(.)$ is a distance (e.g., Bhattacharrya distance[13]) and $o_j, o_k \in O$.

forever do

Find the closest observations (the smallest member of set D, $d_{min} = argminD$) and return the observation indexes (o_{i_1}, o_{i_2}).

if $d_{min} > \alpha$

break

else

Remove observations o_{i_1} and o_{i_2} from the set O, add $o_{i_1 i_2} = \{o_{i_1}, o_{i_2}\}$ to set O.

Remove $d(o_{i_1}, o_j)$ and $d(o_{i_2}, o_j)$ from D, add $d(o_{i_1 i_2}, o_j) = mean(d(o_{i_1}, o_j), d(o_{i_2}, o_j)), 1 \le j \le N$.

Remove indexes i_1 and i_2 from set I, add new index $i_1 i_2$ to I.

endif

B: Remove The Outliers

find μ, the size of the largest subset of set O.

Remove subsets of O whose size is smaller than $\beta\mu$.

C: Fuse Information Within Subgroups

Do for each subgroup

Fuse subgroup observations using POP. Proportional to the maximum distance between subgroup members, choose a value for κ.

end Do

D: Fuse Information Between Subgroup

Using POP, fuse subgroup representatives. Take $\kappa = 1$

Table 3.3: *Algorithm for Grouping observations and removing faulty observations*

The algorithm in Table3.3 can be used to set κ. In general the idea is to find and group the similar observations. The inputs to the algorithm are observations set O and two thresholds α and β $(0 < \beta < 1)$. α is used as a measure to include observation in a subgroup. A number of observations can be included in a subgroup if the distance between each pair of the measurements is less than α. β, is a benchmark for removing small subgroups which are considered as faulty observations or outliers. We consider the subgroup with number of members less than β times numbers of the largest group (after the grouping in the first stage) as outliers. The algorithm is divided into four steps. In the first step, we find the most similar pair of observations and merge them into a subgroup. This step is similar to average linkage [31] in hierarchical clustering. After finding the two closest observations, we remove the observations from the observation set O, group the observations and put them in a set and add the new set to O. The distance between the new set and other members of O is considered as the average distance from the new set members to members of O. We keep repeating this step until no similar observations can be found. In the second step, we remove the outliers. In the third step we fuse the observations in the subsets. For each subgroup a different κ is chosen. The value should be proportional to the maximum distance between the group observations (in the simulation and real world experiment, we used Bhattacharrya distance [13] and took $\kappa = .01\times$ maximum distance). After this stage, we take $\kappa = 1$ and fuse the subgroup representatives using POP. We choose $\kappa = 1$ because the observations are not grouped together and assume there is a disagreement among them. Fig.3.9(b) shows the result of tracking example. This time we used the auto adjusting algorithm to set κ. Comparing to manual setting, a reduction in estimation error is visible.

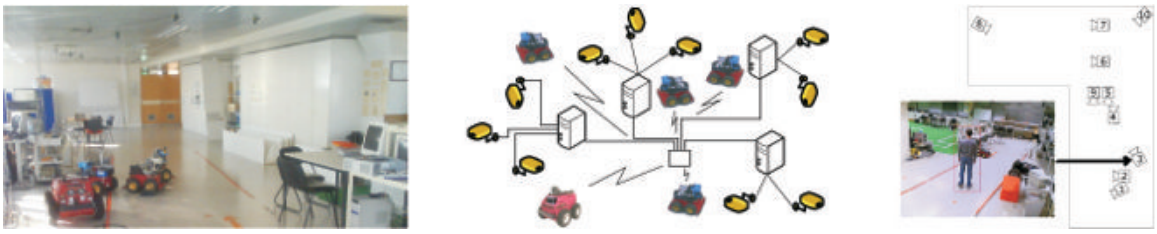


Figure 3.10: *Left: Partial view of the ISROBOTNET testbed, showing most of the cameras in the ceiling and the mobile robots equipped with several sensors. Center: Diagram of the connectivity links between cameras and robots, which have to share the limited network bandwidth. Right: The map of the environment showing the cameras' field-of-view and a picture taken from camera 3. The figure is taken from [10].*

3.5.6 Experiment

In order to evaluate the POP fusion method, we ran one real world experiment. In this experiment, we try to estimate the position of a mobile robot which is manually driven through our lab. In Fig. 3.10 a partial view of the lab and the robot can be seen. In this figure the diagram of connectivity links between cameras and the robots is shown and the position of the cameras are shown. There are 10 ceiling-mounted fixed camera, each partially observes part of the field and has an uncertainty. As example, detail of the uncertainty models of cameras 3 and 8 are shown in Fig. 3.11. Parts of the lab are covered by more than one camera. On the other hand, there are areas which are not covered by any of the cameras.

The cameras are networked. Each camera in the network has a frame rate of 30 fps, and a resolution of 640×480 pixels. A background subtraction algorithm [35] is used to detect the robot positions. A 2-D Gaussian model is considered for the cameras. More

details on the experimental setup are provided in [10]. The robot is equipped with a Sick laser scanner and the laser data is considered as the ground truth for localization of the robot. We use POP in combination with a MCL (Monte Carlo Localization [39]) with 500 particles to track the robot. POP with a fixed κ ($\kappa = 0.1$) is used to fuse the information when more than one camera observes the target. The weighted average particle is chosen as the final estimation.

To compare the results, two methods are chosen as benchmarks: WLC in combination with a standard Kalman filter and a Bayesian method in combination with a MCL. First, we fuse the observations if there are more than one camera observing the robot. Then we use a filter to determine the final estimate. We randomly biased 30% of the cameras detecting the robot (the number of the cameras should be more than 2). In Figs. 3.12(a) and 3.12(b), the results of tracking using Kalman filter and WLC are plotted. To have a better visibility, the robot path is depicted in two different figures. The continuous (green) line represents the

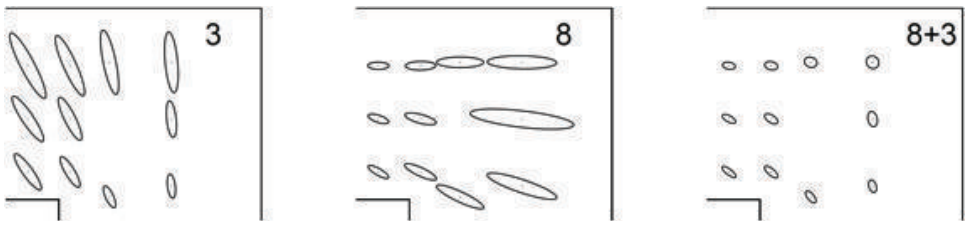
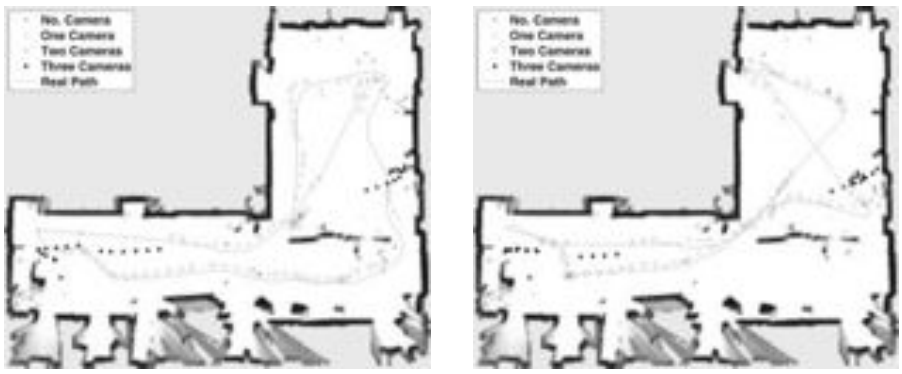


Figure 3.11: *Detail of the uncertainty models of cameras 3,8 and their fusion. Refer to Figure 3.10 for the cameras' locations. The figure is taken from [10].*

robot "real" path (obtained by using a laser scanner on the robot) and markers represent the estimated path. Different markers are used to represent the number of cameras observing the robots for each time instant. For a similar scenario, the result of standard Bayesian in

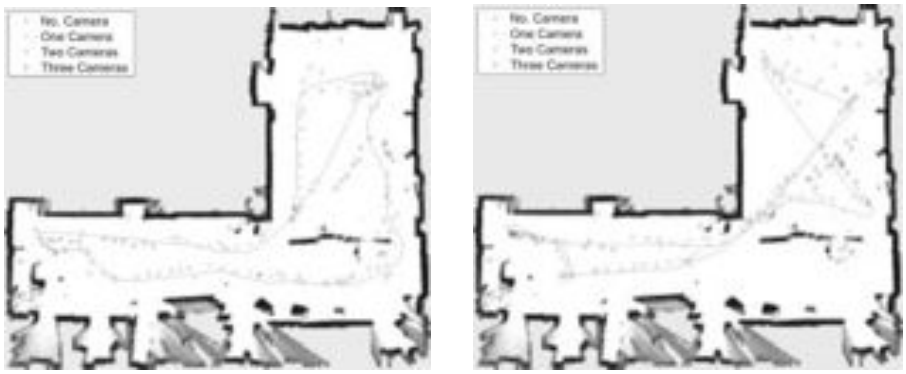


(a) (b)

Figure 3.12: *The figures represents the results of the real experiment. In this experiment we use WLC for fusion and Kalman filter for the tracking. To make the results more visible, the results are drawn into two different figures (left and right figures).*

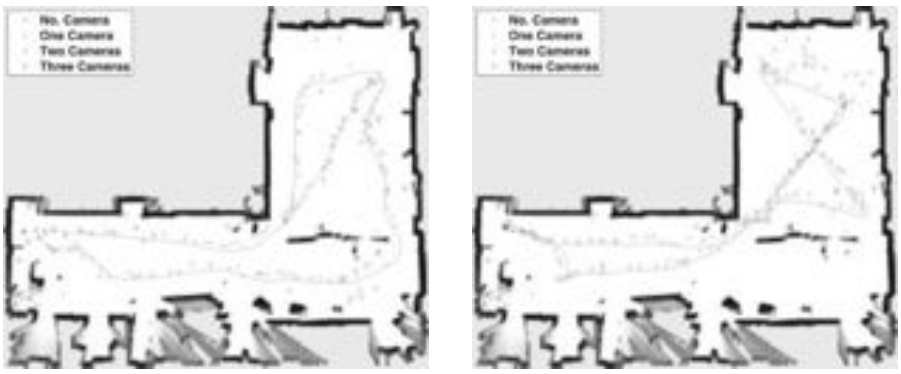
combination with a MCL are shown in Figs.3.13(a) and 3.13(b). POP and MCL results are drawn in Figs. 3.14(a) and 3.14(b). Error histogram of three different methods are compared in Fig.3.15(a). As we can see from the figures, POP and MCL results are much better than the two. On the average, combination of POP and MCL compared to WLC and Kalman filter reduces the estimation error by more than 20% and compared to Bayesian and MCL reduces the estimation error by more than 35%.

As we mentioned above, a fixed κ is used to fuse the information using POP. In 3.5.5, we explain an algorithm for automatically adjusting κ. Now, we evaluate the POP algorithm using the auto adjusting algorithm. Fig. 3.15(b) compares the error histogram auto adjusting and fixed κ. From the figure, we can conclude that POP with auto adjusting κ generates better results than POP with a fixed κ. Figure 3.15(c) represents variation of withing-group κ during the process.



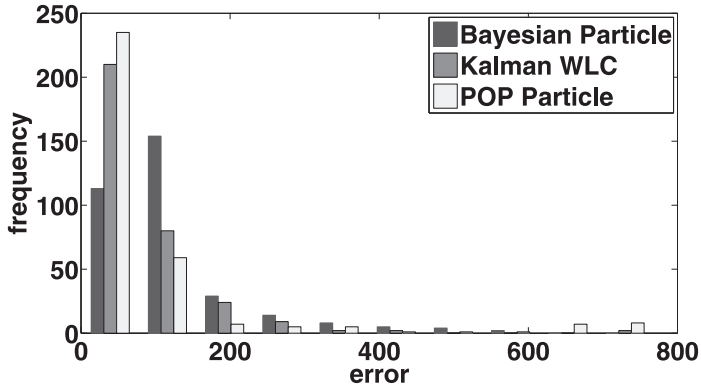
(a) (b)

Figure 3.13: *The figures represents the results of the real experiment. In this experiment we use Bayesian method for the fusion and particle filter for the tracking . To make the results more visible, the results are drawn into two different figures (left and right figures).*

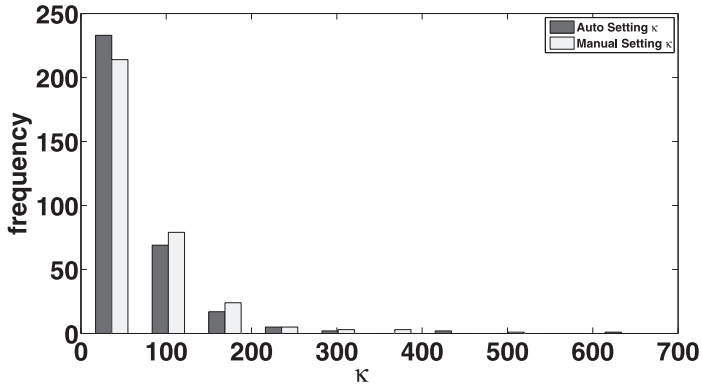


(a) (b)

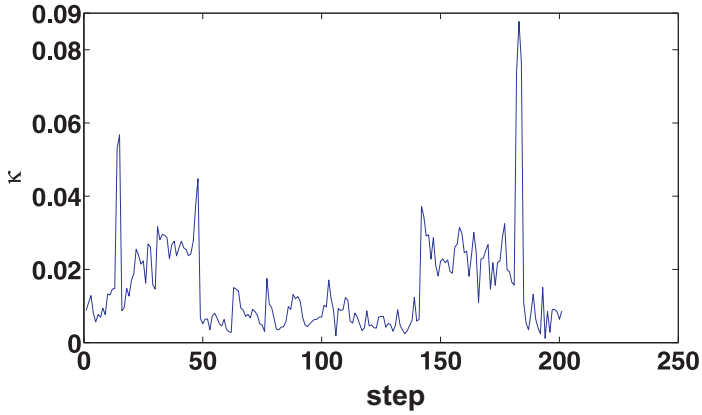
Figure 3.14: *The figures represents the results of the real experiment. In this experiment we use* **POP** *for the fusion and particle filter for the tracking. To make the results more visible, the results are drawn into two different figures (left and right figures).*



(a)



(b)



(c)

Figure 3.15: *The top plot compares the error histogram of the 3 different methods. The middle plot compares the error histogram of* **POP** *with manual and auto adjustment. The bottom plot shows the variation of* κ *when the algorithm Table3.3 is used for automatically adjusting the* κ.

Chapter 4

Active Cooperative Perception

4.1 Introduction

Consider a target tracking scenario in which a number of networked cameras are attempting to localize the position of a target. Some of the cameras are mobile but others are fixed. There are cameras that can change the pan/tilt angles. As mentioned before, the information provided by the cameras is uncertain and contaminated with stochastic noise. One way to reduce the uncertainty in target localization estimation is to fuse the observations of the cameras. The question is, how can we further reduce the uncertainty? One solution can be to add more information sources and apply sensor fusion algorithms (e.g., from Chapter 3). Another solution might be to replace the sensors with higher quality sensors, but we want to avoid this solution since it increases sensor costs. What we are looking for is to use the current setup to improve the observation quality, in a way that improves the results obtained with CP methods.

In general, a sensor provides its best measurements in a specific range which is called the

effective range. Outside of this range, the sensor delivers more uncertain measurements or is not able to measure. Let us assume the sensor uncertainty increases proportionally to the absolute relative distance between the target and the sensor. In our example, if we can adjust the distance between some of the sensors and the target, the target localization uncertainty is reduced for those sensors. Further, if we fuse the cameras information, the target localization uncertainty can be reduced. The idea is that, given the sensor specifications and the current network state, we can perform actions that result in reducing the uncertainty. For instance, if the camera is on the top of a robot, we can adjust the relative distance between the robot and the object of interest by moving the robot closer. If we have a ceiling mounted pan/tilt camera that does not cover the area, we can command the camera to change its field of view. Therefore, a set of actions may exist that results in uncertainty reduction. Those are examples of actions that can be carried out to reduce the uncertainty of the observations. This is designated as Active Cooperative Perception (ACP), because it actively changes sensors positions while implementing what we defined in the previous chapter as CP.

Active Cooperative Perception is a multidisciplinary problem. It is a combination of estimation, cooperative perception and decision making problems. Based on the data acquired from the sources of information, an prior estimate of the state is obtained. To improve the estimation quality index, a set of actions is taken in order to reach a predefined target, e.g., optimizing an objective function.

As we mentioned in 3, CP is the process of establishing cooperation among sensors in order to improve the perception of the whole team. In ACP, the goal is the same as CP. However, we assume that there are actions that can be implemented in order to improve the team perception. The goal is to choose at each step (the set of optimal) action(s). The strategy

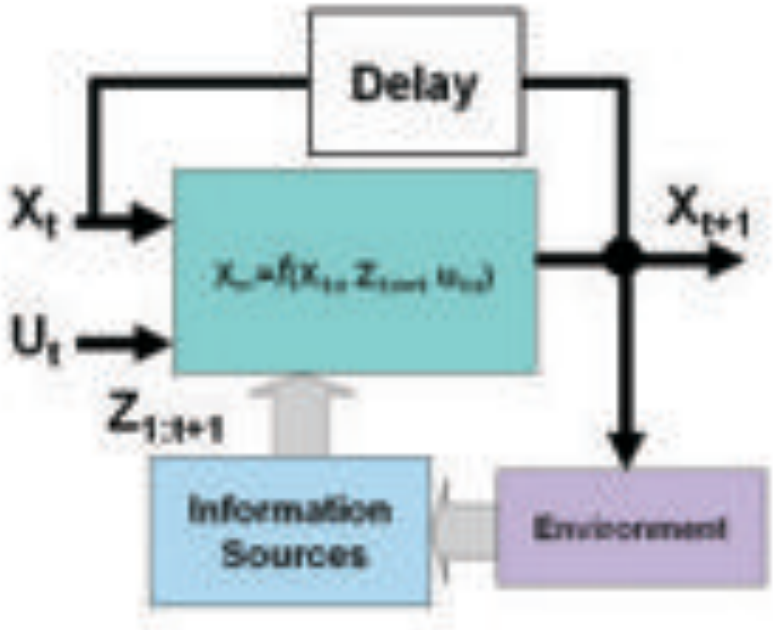


Figure 4.1: *Schematic diagram of an estimator.*

to select actions is based on optimizing a cost function in which the cost and benefit of the actions (improving perception quality) is considered.

4.2 Problem Formulation

Suppose we are interested in estimating the state of the target, modeled as a random variable by using the information provided by a number of sensors. Supposedly, the information sources are contaminated with noise and are uncertain. An estimator determines an approximation of the correct state value so as to reduce some error criteria. It approximates quantities or qualities whose estimates are a function of many known/unknown input factors.

In Fig 4.1, the schematic diagram of an estimator is drawn. As we can see from the figure, the current state estimate of the target x_{t+1}[1] is a function of past states estimate $x_{1:t}$, input commands $u_{1:t}$ and the information $z_{1:t+1}$ provided by the information sources received from the environment:

$$x_{t+1} = f(x_{1:t}, z_{1:t+1}, u_{1:t}) \tag{4.1}$$

The following model:

$$p(x_{t+1}|z_{1:t+1}, x_{1:t}) \tag{4.2}$$

is a probabilistic model of stochastic state evolution. In this model, the prior estimates, control commands and sets of observations are the only information used in order to determine the posterior estimate. This model expresses the stochastic relationship between the past information and the current one.

We need a standard way to measure the amount of uncertainty and compare it with the desired uncertainty level. Hereafter, we call it estimation quality index. Using such an index, we can measure the estimation uncertainty. For example, the error covariance is one well-known measure of estimation quality. This measure is a good index, if the error is normally distributed. Entropy is an alternative measure of estimation quality which is often used for general error distributions.

Based on the above discussion, we associate estimation quality index h with the prior estimate of the random variable. This measure can tell us whether or not any further improvement is required. If the uncertainty level does not meet our needs, according to the ACP definition, we must find a set of actions whose selection would reduce the uncertainty level. After implementing the actions and determining the posterior estimation, the poste-

[1] To simplify the notation, we will usually omit explicit mention of the random variable whenever possible, and instead use the common abbreviation p(x) instead of writing p(X = x).

rior estimation quality index is measured. We expect that the posterior estimation quality index will be better than the prior estimation quality index. Therefore, the objective is to maximize the difference between the estimation quality index before and after implementing the action set A. A is a set whose members are the possible actions for each agent. For instance, in the example of Section 4.1, A can be the next position of the cameras. For a pan-tilt camera it is the next pan and tilt angles while for a mobile camera, it is the next pose of the robot. If h_{pri} and h_{pos} represent the estimation quality index before and after implementing the action set A, then $d(h_{pos}, h_{pri})$ represents the change estimation quality index due to action set A. A metric distance is an example of $d(.)$ function, e.g., the arithmetic difference between the two quality factors. The general idea of the ACP problem is to optimize the objective function by choosing the appropriate action set A.

$$\begin{aligned} A_t^* &= argmax_{A_t} d(h_{pos}, h_{pri}) \\ &subject\ to\ \ h_{pri} > h_{pos} \end{aligned} \tag{4.3}$$

In other words, the goal is to take the action set that optimizes the objective function such that the estimation quality index after implementing action a is improved.

4.3 Assumptions

To simplify the problem, some assumptions are made. As already mentioned, in general the estimation quality index might be improved by performing a sequence of actions. We assume that executing the action set, although this changes the environment state, does not change the target state. If the target state changes due to performing the action, the problem becomes more complex, specially if the change is not deterministic. Another assumption

is about the time needed to perform the action. We assume there is enough time to perform the action and that, while performing the action, the agent state does not change. In general, action execution might not be fast enough, meaning that before completion of the action, the state of the agent might change. In this case the cost function is not optimal anymore and we may not get as much benefit as we would have expected.

We also assume that the sequence of performing the simultaneous actions is not important. Consider a_1 and a_2 as the only two possible actions. Performing action sequence a_1 and then a_2 has the same effect on the quality of the perception as performing a_2 and then a_1.

4.4 A Probabilistic Framework to Active Cooperative Perception

In this section we formalize the ACP problem by using a probabilistic framework. First, we derive a formula for recursively updating the state of interest, e.g., the target position. Later, we derive an optimal action selection mechanism. By employing these two mechanisms, we set up an ACP framework. Having an prior estimate, we calculate the possible action set. After implementing the actions and taking the observations, using the recursive formula, we can update the beliefs. Knowing the prior and posterior estimate qualities, we can decide whether or not any action is required.

4.4.1 Updating the Belief

We use random variables to represent the observations, states and control commands. The objective is to estimate the state of interest, e.g., target position, based on the past states, history of the observations and control commands. We use x_t, z_t, u_t to represent the target's state, observation and control command respectively. Subindex t represents the time. Here we assume that the current state of the system only depends on the current observation, current control command and the previous state (Markovian assumption):

$$p(x_t|z_{1:t}, x_{1:t-1}, u_{1:t}) = p(x_t|z_t, x_{t-1}, u_t) \tag{4.4}$$

Based on the Markovian assumption:

$$p(z_t|z_{1:t-1}, x_{1:t}, u_{1:t}) = p(z_t|x_t) \tag{4.5}$$

Using (4.4), we can also conclude:

$$p(x_t|z_{1:t-1}, x_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \tag{4.6}$$

One of the consequences of this assumption is that, to update the states, we need less computational effort. Also we need less memory because we do not need to keep all the past information. Based on these assumptions, we can determine the evolution of state:

$$\begin{aligned} p(x_t|z_{1:t}, u_{1:t}) = \frac{p(z_t, x_t, z_{1:t-1}, u_{1:t})}{p(z_t, z_{1:t-1}, u_{1:t})} = \frac{p(z_t, x_t|z_{1:t-1}, u_{1:t})p(z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})p(z_{1:t-1}, u_{1:t})} = \\ \frac{p(z_t, x_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})} = \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})} \end{aligned} \tag{4.7}$$

We replace the denominator by normalization constant $\frac{1}{\eta}$.

$$p(x_t|z_{1:t}, u_{1:t}) = \eta\, p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t}) \tag{4.8}$$

By substituting (4.5) in (4.8):

$$p(x_t|z_{1:t},u_{1:t}) = \eta\, p(z_t|x_t)p(x_t|z_{1:t-1},u_{1:t}) \tag{4.9}$$

The second term in the numerator of equation (4.7) can be written as:

$$p(x_t|z_{1:t-1},u_{1:t}) = \int p(x_t,x_{t-1}|z_{1:t-1},u_{1:t})d_{x_{t-1}} = \int p(x_t|x_{t-1},u_t)p(x_{t-1}|z_{1:t-1},u_{1:t-1})d_{x_{t-1}} \tag{4.10}$$

Combing (4.9) and (4.10) we can conclude:

$$p(x_t|z_{1:t},u_{1:t}) = \eta\, p(z_t|x_t)\int p(x_t|x_{t-1},u_t)p(x_{t-1}|z_{1:t-1},u_{1:t-1})d_{x_{t-1}} \tag{4.11}$$

The above equation recursively determines the prior estimate of the state. It represents the probability of being in a particular state at time t given the transition probability $p(x_t|x_{t-1},u_t)$, observation model $p(z_t|x_t)$ and the prior probability $p(x_{t-1}|z_{1:t-1},u_{1:t-1})$.

Knowing the prior state estimate, a series of actions can be performed to improve the estimation quality index. Here, we assume that the actions are applied sequentially and we relate to each action a_t an observation o_t where subindex t indicates the time.

To simplify the notation, we denote the prior estimate of the system $p(x_t|z_{1:t},u_{1:t})$ as $p(y_t)$:

$$p(y_t) = p(x_t|z_{1:t},u_{1:t}) \tag{4.12}$$

Therefore, we have a prior knowledge about state of the system which is shown by $p(y_t)$. Now, we perform a series of action $a_{t+1},a_{t+2},...,a_{t'}$ and we observe $o_{t+1},o_{t+2},...,o_{t'}$. We start from time t and assume target state does not change between times t and t'. First, we implement action a_{t+1} and observe o_{t+1}. After updating the state and measuring the estimation quality index, we decide if we need to perform any further action. The update can be done using Bayes' rule. If we need to implement another action, the updated belief

plays the role of a prior for the next update. Let's assume the final estimate after performing actions $a_{t+1}, a_{t+2}, ..., a_{t'}$ and observing $o_{t+1}, o_{t+2}, ..., o_{t'}$ is called $y_{t'}$. Using Bayes' rule, we can determine the update formula:

$$\begin{aligned} p(y_{t'}|a_{t+1:t'}, o_{t+1:t'}) &= \frac{p(o_{t'}, y_{t'}, a_{t+1:t'}, o_{t+1:t'-1})}{p(a_{t+1:t'}, o_{t+1:t'})} \\ &= \frac{p(o_{t'}|y_{t'}, a_{t+1:t'}, o_{t+1:t'-1}) p(y_{t'}, a_{t+1:t'}, o_{t+1:t'-1})}{p(o_{t'}, a_{t+1:t'}, o_{t+1:t'-1})} \\ &= \frac{p(o_{t'}|y_{t'}, a_{t+1:t'}, o_{t+1:t'-1}) p(y_{t'}, a_{t+1:t'}, o_{t+1:t'-1})}{p(o_{t'}|a_{t+1:t'}, o_{t+1:t'-1}) p(a_{t+1:t'}, o_{t+1:t'-1})} \\ &= \frac{p(o_{t'}|y_{t'}, a_{t+1:t'}, o_{t+1:t'-1}) p(y_{t'}|a_{t+1:t'}, o_{t+1:t'-1}) p(a_{t+1:t'}, o_{t+1:t'-1})}{p(o_{t'}|a_{t+1:t'}, o_{t+1:t'-1}) p(a_{t+1:t'}, o_{t+1:t'-1})} \\ &= \frac{p(o_{t'}|y_{t'}, a_{t+1:t'}, o_{t+1:t'-1}) p(y_{t'}|a_{t+1:t'}, o_{t+1:t'-1})}{p(o_{t'}|a_{t+1:t'}, o_{t+1:t'-1})} \end{aligned} \tag{4.13}$$

We can write, due to Markov assumption:

$$p(o_i|y_i, a_{1:i}, o_{1:i-1}) = p(o_i|y_i, a_i) \tag{4.14}$$

In other words, each observation o_i depends only on the most recent action a_i and target state. Other actions and observations have no influence on it. It is obvious that o_i is independent of a_{i-1}. We also assume that o_i is independent of previous observation.

Also we can write:

$$p(y_i|a_{1:i}, o_{1:i-1}) = p(y_i|a_{1:i-1}, o_{1:i-1}) \tag{4.15}$$

The reason is as follows: states are unobservable and we only estimate them from observations. When we only perform an action, the observed result of the action, provides some information about the state. Without observing o_i, a_i does not provide any information about the state. We also replace the denominator (4.13) with the normalization constant $\frac{1}{\eta}$. All together, we can simplify (4.13) as:

$$p(y_{t'}|a_{t+1:t'}, o_{t+1:t'}) = \eta\, p(o_{t'}|y_{t'}, a_{t'}) p(y_{t'}|a_{t+1:t'-1}, o_{t+1:t'-1}) \tag{4.16}$$

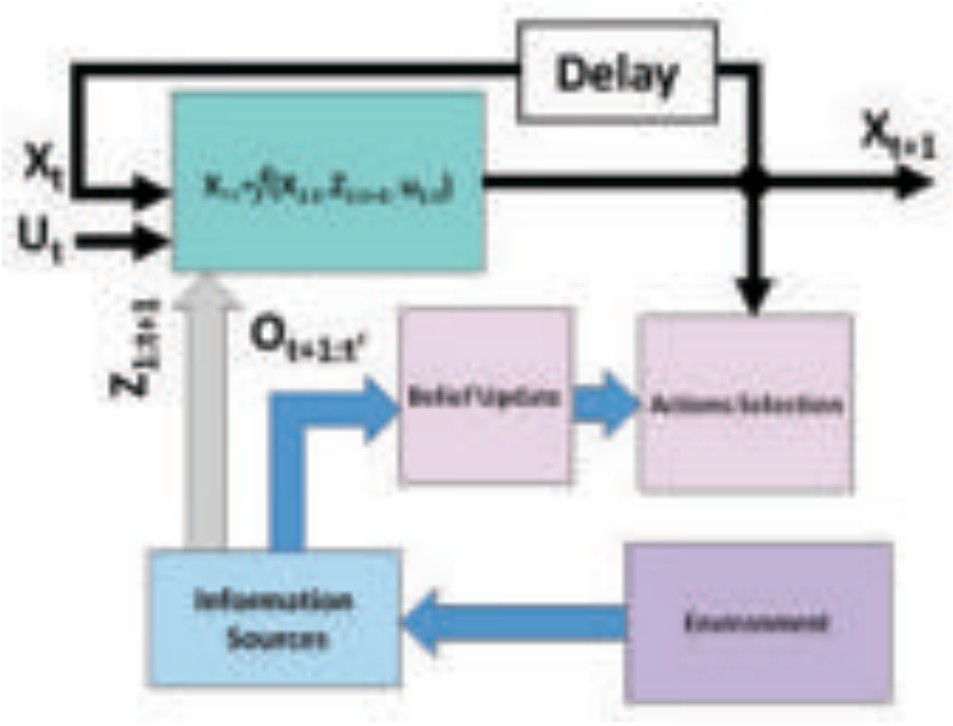


Figure 4.2: *Schematic diagram of an ACP system.*

Therefore, (4.16) represents a recursive relationship between prior and posterior beliefs. By determining the likelihood $p(o_{t'}|y_{t'}, a_{t'})$, the prior belief can be updated to the posterior belief. Term $p(y_{t'}|a_{t+1:t'-1}, o_{t+1:t'-1})$ in equation (4.16) represents the prior belief. Term $p(o_{t'}|y_{t'}, a_{t'})$ is a likelihood that should be determined in advance. A schematic diagram of an ACP system is drawn in Fig.4.2. As we can see from the figure, at each time step, first a prior estimate x_{t+1} of the target using the histories of sensory information $z_{1:t+1}$ and control command $u_{1:t+1}$ is determined. Based on the estimate and if the prior estimate x_{t+1} is not certain enough, the system selects (equation (4.3)) an optimal action set $a_{t+1:t'}$, the action set that reduces the uncertainty the most. At time t' we end the process because either the observation uncertainty is small enough or cannot be improved any further. In

the action selection block, first, the uncertainty of the current estimation is measured. If the uncertainty is not small enough, then, we choose the optimal action set. After implementing the action set, through the information sources we receive the new information from the environment. In the belief update block, by using the new information, the previous belief is updated. Now, we have a better estimation. Again in action selection block, we measure the uncertainty content of the belief and if it is not at the desire level, we choose another action. We continue the process until there is no action set that can improve the belief. The process starts from time t+1 and ends at time t'.

4.4.2 Using Entropy as Estimation Quality Index

As we explained before, we associate an estimation quality index h with the prior belief. The estimation quality index measures the amount of uncertainty in the estimation. Here, we use the entropy to measure the estimation quality index. If the estimation is certain enough, e.g., the entropy of prior estimation is lower than a predefined threshold or implementing the action does not improve the estimation quality index, no more actions are required. Based on this definition, the amount of uncertainty in the prior estimate can be calculated as follows:

$$H(p(x_t|z_{1:t}, u_{1:t}) = -\int p(x_t|z_{1:t-1}, u_{1:t}) \log p(x_t|z_{1:t-1}, u_{1:t}) d_{x_t} \tag{4.17}$$

where H stands for the entropy.

Assuming that the current estimation is not certain enough:

$$H(x_t|z_{1:t-1}, u_{1:t}) \geq \lambda \tag{4.18}$$

(where λ is a predefined threshold), we need to increase the level of estimation quality index. This might be done by performing a set of actions such that the maximum improvement in amount of certainty can be obtained. The overall estimation quality index should be increased.

The estimation quality index of the posterior estimate is:

$$H(x_t|z_{1:t},u_{1:t},a_t,o_t) = -\int p(x_t|z_{1:t},u_{1:t},a_t,o_t)\log p(x_t|z_{1:t},u_{1:t},a_t,o_t)d_{x_t} \tag{4.19}$$

Now we have to determine the amount of change. Here, we calculate the difference between the estimation quality index before and after implementing the actions sequence:

$$I_H = H(p(x_t|z_{1:t},u_{1:t},x_{1:t-1})) - H(p(x_t|z_{1:t},u_{1:t},x_{1:t-1},a_t,o_t)) \tag{4.20}$$

As we can see, I_H is a function of action set a. Actually in this formula a plays an important role. By choosing different action sets, we can change I_H.

$$a_t^* = argmax_{a_t} I_H \tag{4.21}$$

The main objective is to reduce uncertainty. The optimal I_H tells us which action set should be chosen in order to reduce the uncertainty the most.

I_H in equation (4.20) tells us how much information we will gain after performing the action set. This value is known as mutual information. For two continuous random variable A and B:

$$I(A;B) = H(A) - H(A|B) = \int_A \int_B p(a,b)\log\frac{p(a,b)}{p(a)p(b)} \tag{4.22}$$

where $p(a,b)$ is the joint probability distribution function of A and B, and $p(a)$ and $p(b)$ are the marginal probability distribution functions of A and B respectively. Again for simplicity,

we use notation $p(y_t|o_t,a_t)$ instead of $p(x_t|z_{1:t},u_{1:t},x_{1:t-1},o_t,a_t)$.

Observation o_t depends on the action set a_t:

$$p(o_t|a_t)=\int_{x_t}p(o_t,x_t|a_t)dx_t=\int_{x_t}p(o_t|x_t,a_t)p(x_t|a_t)dx_t=\int_{x_t}p(o_t|x_t,a_t)p(x_t)dx_t \quad (4.23)$$

Again, if we define $p(y_t)=p(x_t|z_{1:t},u_{1:t},x_{1:t-1})$, by using equation (4.22), we can write:

$$I(y_t;o_t,a_t)=I(y_t;o_t|a_t)=H(y_t)-H(y_t|o_t,a_t) \quad (4.24)$$

$$I(y_t;o_t|a_t)=\int_{y_t}\int_{o_t}p(y_t,o_t|a_t)\log\frac{p(y_t,o_t|a_t)}{p(y_t)p(o_t|a_t)}do_tdy_t \quad (4.25)$$

$$I(y_t;o_t|a_t)=\int_{y_t}\int_{o_t}p(o_t|y_t,a_t)p(y_t|a_t)\log\frac{p(o_t|y_t,a_t)p(y_t|a_t)}{p(y_t)p(o_t|a_t)}do_tdy_t \quad (4.26)$$

Using $p(y_t|a_t)=p(y_t)$ from (4.26) we can write:

$$I(y_t;o_t,a_t)=\int_{y_t}p(y_t)\int_{o_t}p(o_t|y_t,a_t)log\frac{p(o_t|y_t,a_t)}{p(o_t|a_t)})do_tdy_t \quad (4.27)$$

Combining (4.23) and (4.27) we will have:

$$I(y_t;o_t,a_t)=\int_{y_t}p(y_t)\int_{o_t}p(o_t|y_t,a_t)log(\frac{p(o_t|y_t,a_t)}{\int_{y_t}p(o_t|y_t,a_t)p(y_t)dy_t})do_tdy_t \quad (4.28)$$

As we can see from (4.13) and (4.28), for calculating both mutual information and also updating state we only need to calculate $p(o_t|y_t,a_t)$. This likelihood function needs to be learned in advance or calculated on line. In later sections, we will discuss the ways to tackle this problem. ACP algorithm in pseudo-code is depicted in Table 4.1.

4.5 Active Cooperative Perception For a Linear System With Gaussian Uncertainties

Driving a closed form formula for ACP in general is not possible. Here we consider linear systems with Gaussian noise. These assumptions are made to simplify the calculation and

ACP Algorithm ***(prior belief*** $p(x_t|z_{1:t}, u_{1:t})$***)***

Do forever

Choose optimal action set A using (4.21)

if the action set A is not empty

Perform the optimal actions.

Retrieve the new information.

Update the belief using (4.16)

end

if the uncertainty of belief is small enough (use 4.18)

break.

end

end forever

Table 4.1: *ACP Algorithm*

derive a closed form formula for ACP. As a result of this we can improve the speed of the algorithm and make it suitable to apply to a team of robots. State update can be done using a closed form formula. Also, by using those assumptions, a closed form formula can be derived for the action selection step. Here, we show how to use these two tools together to make ACP problem easier to manage.

Consider a target whose state evolution can be expressed as below:

$$x_{k+1} = f_k(x_k) + v_k \tag{4.29}$$

where $x_k \in \mathbb{R}^n$, n is the dimension of the state vector and x_k is the state vector of the system and $k \in \mathbb{N}$ is the time index. f_k is a known function and v_k is called the process

noise which accounts for effects of unforeseen disturbances and parameter mismatch in the model. Using the above equation we can recursively predict the evolution of the state space. Furthermore, if some measurement z_k is available, we can use this information to filter the predicted state.

$$z_{k+1} = g_k(x_{k+1}) + w_{k+1} \tag{4.30}$$

where $z_{k+1} \in \mathbb{R}^m$, m is the dimension of the measurement vector and z_k is the measurement, g_k is a known function and w_{k+1} is the measurement noise, including effects of unforeseen disturbances in the modeling of the sensor. Actually this equation incorporates the sensors information into the model.

In passive systems, the observation model of the sensors only depends on the current state of the system. On the other hand, in a system where there is a possibility of changing the observation model, equation (4.30) is also a function of the action that selects the sensor(s).

$$z_{k+1} = g_k(x_{k+1}, a_{k+1}) + w_{k+1} \tag{4.31}$$

a_{k+1} represents all the parameters that can affect the observation model.

A special case of this system is a linear system where f and g are linear function of x:

$$x_{k+1} = Ax_k + v_k \tag{4.32}$$

$$z_{k+1} = C(a_{k+1})x_{k+1} + w_{k+1} \tag{4.33}$$

In this section, we consider the ACP framework for linear systems, assuming that all the uncertainties and noise are Normally distributed.

As we know, the Kalman Filter (KF) can obtain the optimal state estimate for this class of systems. The KF is a sequential minimum MSE (Mean Squared Error) estimator and is optimal in a minimum MSE sense if the uncertainties are Gaussian. When f and g are not

linear we can use the Extended Kalman Filter to tackle the problem.

KF provides an answer that have the minimum posterior covariance. Knowing that the system is linear and the priors and noises are Gaussian, the following relationship between the prior and posterior estimations can expressed as:

$$p(x_k|z_k) = \mathcal{N}(x_k; x_{k|k}, P_{k|k}) \tag{4.34}$$

$$p(x_{k+1}|z_k) = \mathcal{N}(x_k; x_{k+1|k}, P_{k+1|k}) \tag{4.35}$$

$$p(x_{k+1}|z_{k+1}) = \mathcal{N}(x_{k+1}; x_{k+1|k+1}, P_{k+1|k+1}) \tag{4.36}$$

where $\mathcal{N}(x; \mu, \Sigma)$ represents Gaussian density function x having mean μ and covariance Σ. After solving the Kalman filter equations, the means and variances of the estimates can be calculated as:

$$x_{k+1|k} = A_k x_{k|k} \tag{4.37}$$

$$P_{k+1|k} = Q_k + A_k P_{k|k} A_k^T \tag{4.38}$$

$$x_{k+1|k+1} = x_{k+1|k} + K_{k+1}(z_{k+1} - C_{k+1} x_{k+1|k}) \tag{4.39}$$

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1} S_{k+1} K_{k+1}^T \tag{4.40}$$

$$S_{k+1} = C_{k+1} P_{k+1|k} C_{k+1}^T + R_{k+1} \tag{4.41}$$

$$K_{k+1} = P_{k+1|k} C_{k+1}^T S_{k+1}^{-1} \tag{4.42}$$

$$P_{k+1|k+1} = [I - (P_{k+1|k} C_{k+1}^T S_{k+1}^{-1}) C_{k+1}] P_{k+1|k} \tag{4.43}$$

where:

$$P_{k+1|k} = Q_k + A_k P_{k|k} A_k^T \tag{4.44}$$

$$S_{k+1} = C_{k+1} P_{k+1|k} C_{k+1}^T + R_{k+1} \tag{4.45}$$

In fact, action a_t can change the covariance of observations noise R. Looking at the equations through, this can change innovation term and as a result of this covariance of posterior distribution will change:

$$P_{k+1|k+1}(a_{k+1}) = [I - (P_{k+1|k}C_{k+1}^T S_{k+1}^{-1})C_{k+1}]P_{k+1|k} \tag{4.46}$$

Also the posterior probability will change:

$$p(x_{k+1}|z_{k+1}) = \mathcal{N}(x_{k+1}; x_{k+1|k+1}, P_{k+1|k+1}(a_{k+1})) \tag{4.47}$$

This is the result of variation of observation model as a function of the action.

Up to now we have concluded that action sequence a_t affects the posterior covariance matrix $P_{k+1|k+1}$. Now the problem is to the find the action set that minimizes the posterior covariance:

$$a_t^* = argmin_a(|P_{k+1|k+1}(a_{k+1})|) \tag{4.48}$$

This can be formulated as an optimization problem. Convex Optimization is a special class of the optimization problems where our objective function has special properties. Although this is a small class of optimization problems, it has received considerable attention due to its special properties.

A convex optimization problem can be defined as:

$$\begin{aligned} &minimize\ f(x) \\ &subject\ to\ \ g_i(x) \qquad 0 \leq i \leq m \end{aligned} \tag{4.49}$$

where f is the objective function and g_i is the set of constraints and both f and g are convex. The first term in the objective function is about action costs and benefits. If there is any cost or benefit, they can be added to the objective function. As we mentioned earlier there might be a set of constraints. Any constraint should be added to the constraint sets. Fortunately if the objective function is convex, there are many algorithms to handle the problem.

4.5.1 Practical Issues

Up to now, we have stated our strategy for recursively updating the estimates and selection of the optimal action. Taking another look at equation (4.16) and assuming the prior estimate is given, we only need to evaluate the term $p(o_{t'}|y_{t'}, a_{t'})$. This term is a likelihood function that can be evaluated through a training period. One way is to model this likelihood function with a parametric probability distribution function, e.g., a Gaussian, and estimate the parameters during the training. To have some insight about the complexity of estimating this function, if we have N states and M possible actions, then we need to learn N.M different densities.

In case of a non-specific distribution, we can model the distribution with a mixture of Gaussian or use Parzen windows [64]. In the discrete case this likelihood function can be learned using different learning algorithms such as neural networks or support vector machines.

Another term which needs to be calculated is $I(y_t; o_t, a_t)$. Let us consider a discrete state space. In (4.28) term $p(y_t)$ is the prior estimate which can be determined from previous step. The only remaining term is $p(o_t|y_t, a_t)$. $I(y_t; o_t, a_t)$ can be estimated using Monte Carlo algorithm. Another way is to mathematically approximate this term which can be used in a discrete state space as well.

4.5.2 Action Cost

So far we have not considered any cost for the actions. Let us to clear this up by introducing a simple example. Consider a scenario shown in Fig.4.3 in which a robot can take different paths $L1, L2$ and $L3$ to approach object O and find out the position of the object. The object is in area where it can only be detected by camera 2. There is another camera, camera

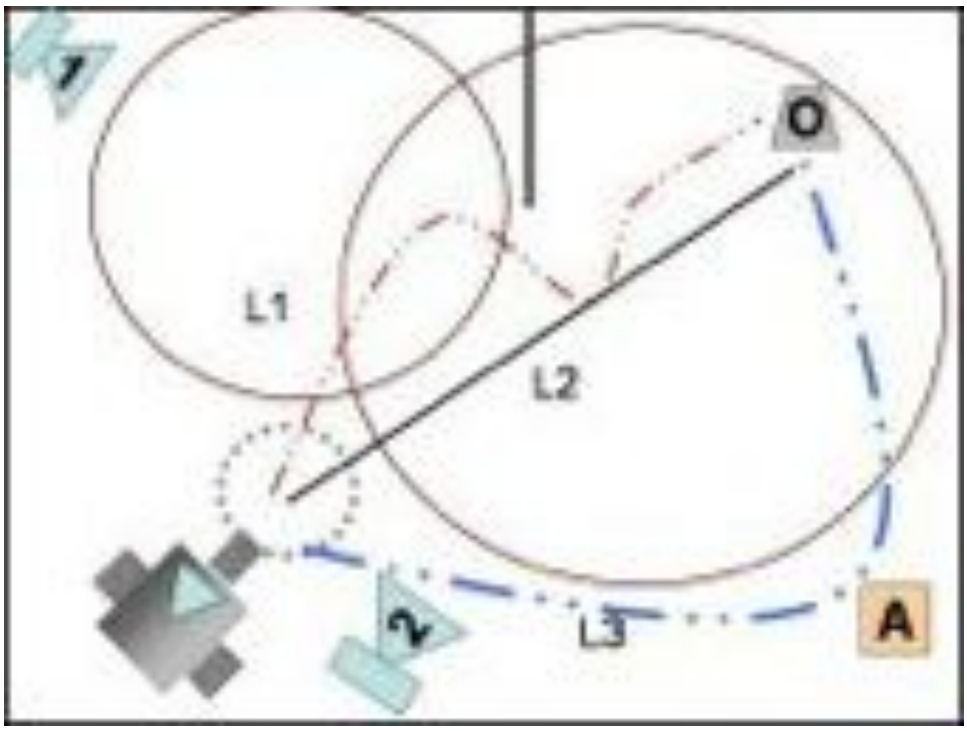


Figure 4.3: *Scenario where a mobile robot can take several paths.*

1. The information of this camera can be used by robot to improve the estimation quality index. This information may be about the objects positions in the field of view as well as the robot. All the sensors are uncertain and we need to know the position of the object as certain as possible. All the information is routed to a central processing unit for further process. If we limit the possible paths to L1, L2 and L3, the shortest path is L2. Therefore the robot needs less energy to reach the target if it takes L2. L2 is also the fastest path, comparing to L1 or L3 paths because it is a straight line. Time and energy resources are two important factors for our scenario and we have to consider them in the calculations. Although in our example time and energy needed for taking path L2 are the lowest, it may not be the optimal path. In other words, there are other important factors that should be taken into

account. We may take a path which is the shortest but because of different obstacles placed on the path and maneuvers needed to avoid obstacles, it may need more time to reach the target comparing to other options. In general, resources limitations impose actions costs. In the above example, time and energy are considered as two examples of the resources. Sometimes, performing an action is more risky than performing others. As an example, consider the same scenario but this time consider path L1. As it can be seen from the figure, there is an obstacle nearby the path; a wall that needs to be avoided. This path is more risky than L2 and L3, because there is the possibility of the robot bumping into the wall that should be considered as a cost for the path. Naturally, the robot has to turn on its obstacle avoidance sensors and does some maneuvers which need energy and time resources and may also cause the robot becomes less certain about its own position. However the robot can be benefited from taking this path comparing to L2 by being observed by camera 1 which improves the estimation quality index of robot localization. Path L3 is longer than path L2 and less risky than L1 but there is object A on the path. The robot will be rewarded if this object is recognized.

Those are examples of the actions costs but certainly the action costs are not limited to these items. In general, we have to consider all possible costs and rewards. Not only we should try to maximize the benefits of the action which previously presented in the terms of difference between uncertainty of observation before and after implementing the action, but also the costs of actions need to be considered.

If we denote the cost function as J(a), the goasl is to determine the action that optimizes J(a):

$$a_t^* = argmax_{a_t}((1-\beta)I(y_t;o_t|a_t) - \lambda\beta J(a_t)) \tag{4.50}$$

The coefficient λ is used to uniformize units of the two terms. β is used to weight one term over the other one and $0 \leq \beta \leq 1$.

4.6 An Illustrative Example

In this section we run a series of the simulations to become more familiar with the idea. First, we implement a simulation of a 1D environment in which a robot and a pan-tilt camera cooperatively localize an object. Next, we will consider a 2D environment. The difference with respect to the previous simulation is the possibility of fusing the observations since in some instants, more than one camera can observe the object of interest. Still we use the same algorithm and basic principles to implement ACP. Before explaining the simulations and introducing the results, we explain the ACP algorithm for a tracking problem.

4.6.1 ACP Algorithm for a Tracking Problem

We consider a scenario in which a robot is tracked by a ceiling mounted pan-tilt camera. The robot tries to approach a target using its own sensory information received (a fixed camera and the odometry sensor) and also the information received from the pan-tilt camera. We use a centralized approach to process the data received from different sources. The system is assumed to be linear and the noise is normally distributed. Therefore a Kalman filter is used to estimate the states. We used the equation derived in Section 4.5. The ACP algorithm in pseudo-code adapted for the example is depicted in Table 4.2. Starting with the prior belief, we calculate the best possible actions for the robots and the cameras. This is done by optimizing the objective function with regard to the constraints set. The output

Algorithm ACP ***(target position, robot position and pan-tilt camera position prior beliefs)***

Do forever

1.choose optimal action considering the prior beliefs of robot, target, pan-tilt cameras and all costs involved.

2.Perform the selected actions.

3.Update robot position belief using the Kalman filter, considering odometry and vision information received from the cameras.

4.Update pan-tilt camera position belief using the Kalman filter.

5.Update the target position belief using vision information. If there is more than one camera observing the target, use WLC rule to fuse the observations.

6.Let the target move

7.Based on the current beliefs of the robot, target and pan-tilt cameras, calculate the new belief of the target position.

end forever

Table 4.2: *ACP Algorithm adapted for the illustrate example*

of this step is a set of actions for the robot and also for the pan-tilt cameras. Line 1 in the table explains this step. In line 2, we perform the actions generated in previous step. For the robot, we use KF to update the estimate of the robot position model. For this update, we always have access to the odometry information. If at least one camera could observe the robot, we can use this information to filter the belief obtained in the KF prediction phase. Since we have assumed that the prior beliefs and sensors noise models are Gaussian and the system is linear, therefore, the posterior belief of robot position is also Gaussian. This update is considered in line 3. In line 4, we use pan-tilt camera information and robot camera to update the belief of target position. The robot and cameras have no access to the target dynamic and cameras are the only available information sources. It is possible that none of the cameras could observe the target. In this case, the prior and posterior beliefs

are the same. The reason is clear: we have no new information to update our belief. If more than one measurement is available, we use the WLC method to fuse the observations. In the next step, we use dynamics of pan-tilt camera to update their current position. By now, the optimal actions are implemented. Therefore, we are able to estimate the position of the target. Now, we take new measurements and estimate the new target position based on the information obtained from the pan-tilt and robot cameras. This new information plays the role of prior estimate for the optimization step in the next cycle of the algorithm. Here, we conduct a simulation to evaluate the performance of the proposed method as well as the impact of taking action on uncertainty. In this simulation, a robot is deployed to track the position of a target. Our goal is to estimate the position of the object with minimum possible uncertainty. Below we explain the components involved in the simulations in more details.

4.6.1.1 The Goal

The simulation goal is to localize the target with minimum uncertainty. This can not be done unless the robot knows its own position with minimum uncertainty. The reason is that, comparing to the tilt camera, the robot can get closer to the object and the observation model is considered in such a way that the amount of uncertainty is proportional to the relative distance between camera and the object. If the robot is uncertain about its own position, its information can not be helpful to localize the target.

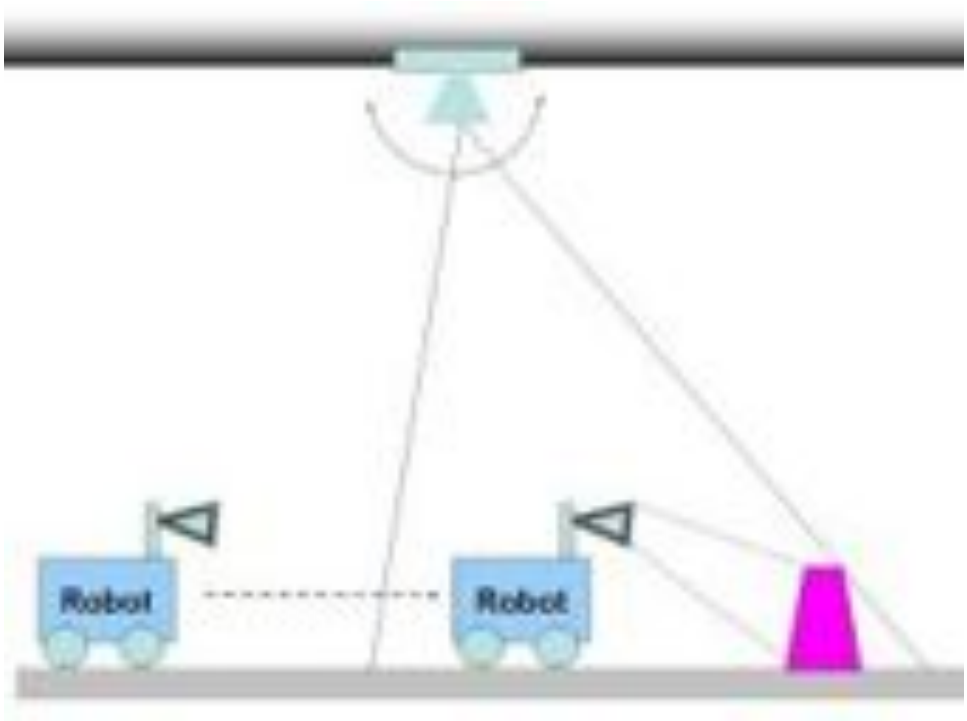


Figure 4.4: *Schematic diagram of the first simulation.*

4.6.2 Simulation: Object Tracking Using a mobile Robot and a Camera

4.6.2.1 Environment

A schematic of the environment is drawn in Fig.4.4 in which the positions of the robot, target and the camera are shown. The tilt camera is mounted in the middle such that it can cover all the environment. We consider a 1D environment and divide it into 100 equal segments.

4.6.2.2 Robot

The robot is equipped with two types of sensors: odometry and vision. Both of them are uncertain and their observations contain additive Gaussian noise. They are considered unbiased sensors. In order to prevent excessive error, we periodically reset the odometry sensor. The variance of the odometry sensor increases quadratically proportional to the traveled distance. The error variance of the object estimate obtained increases as the distance to the object increases. Again it is considered to have a quadratic form but the camera range is limited. Outside of the range the camera is not able to observe any object.

4.6.2.3 Object

There is an object involved in the simulation which is called a target. It is assumed that we have no information regarding the dynamics of the object. The only sources of information about the object position are tilt and robot cameras.

4.6.2.4 Tilt cameras

Since the robot camera has a limited range, a tilt camera is involved in the simulation. It is installed in the middle of the environment. The sensors are unbiased. The noise has a Gaussian form and the error variance increases quadratically with the relative distance between the camera and the object. The tilt camera can change its angle from 45 to 135 degrees.

4.6.2.5 Assumptions

In this simulation, we assume that all the uncertainties have Gaussian form. It is also assumed that all sensors used in this simulation are unbiased.

We assume that initially we have information regarding the positions of the robot, target and the tilt camera. We also assume that the target state does not change during the duration of executing an action.

4.6.2.6 Constraints

In real world scenarios we have limitations. Here, we consider those limitations as constraints, e.g., the angular velocity of a tilt camera or velocity of a robot are limited to some values.

The maximum change in the tilt angles is 5 degrees. The robot velocity is limited to 5 units per second.

To avoid collision between the robot and the target, a minimum relative distance between the robot and target is considered. Although the movement of the target is considered random its position is always inside the environment.

Since the robot camera is fixed, it can observe only the area in front of itself. Therefore the position of the robot is considered behind the object.

4.6.2.7 The Objective Function

As we mentioned earlier, when noise has Gaussian form, the uncertainty can be represented by an error covariance matrix. We consider the objective function as the sum of the

entropies and do not consider any other cost for implementing the actions. Since uncertainty of the sensors are considered to have a quadratic form, the objective function has a quadratic form too.

4.6.2.8 Simulation Results

Here, we discuss the results of the simulation and make our conclusions. The robot real and estimated positions are compared in Fig. 4.5(a). As we can see from the figure, for the first few steps, the only source of information about the position of the robot is the odometry sensor. The tilt camera can not observe the robot. The ACP algorithm forces the tilt camera to turn towards the robot. After the tilt camera finds the robot, it adjusts its angular position by considering the robot and object positions. After we have the two sensors, odometry and tilt camera, the Kalman Filter is used to estimate the robot position. The error between the real position and estimated position is due to the uncertainty of the sensors.

Figure 4.5(b) shows real and estimated paths of the object. Initially, the object is observed by the tilt camera. However, the relative distance between the tilt camera and the object is relatively large, so the observation has large uncertainty. Later the tilt camera is forced to turn towards the robot. Therefore, we lose the object and keep the latest position of the object as its current position. The robot moves towards the object and in epoch 29, the two cameras simultaneously detect the object. When the two cameras observe the object simultaneously, the WLC algorithm is used to fuse the two measurements. From that moment on we could track the object and the error is relatively low.

Our final goal is to reduce uncertainty of object position. As we mentioned before, the uncertainty of the robot localization should be taken into account because it is the robot that can approach the target and track the object. Due to its large relative distance, obser-

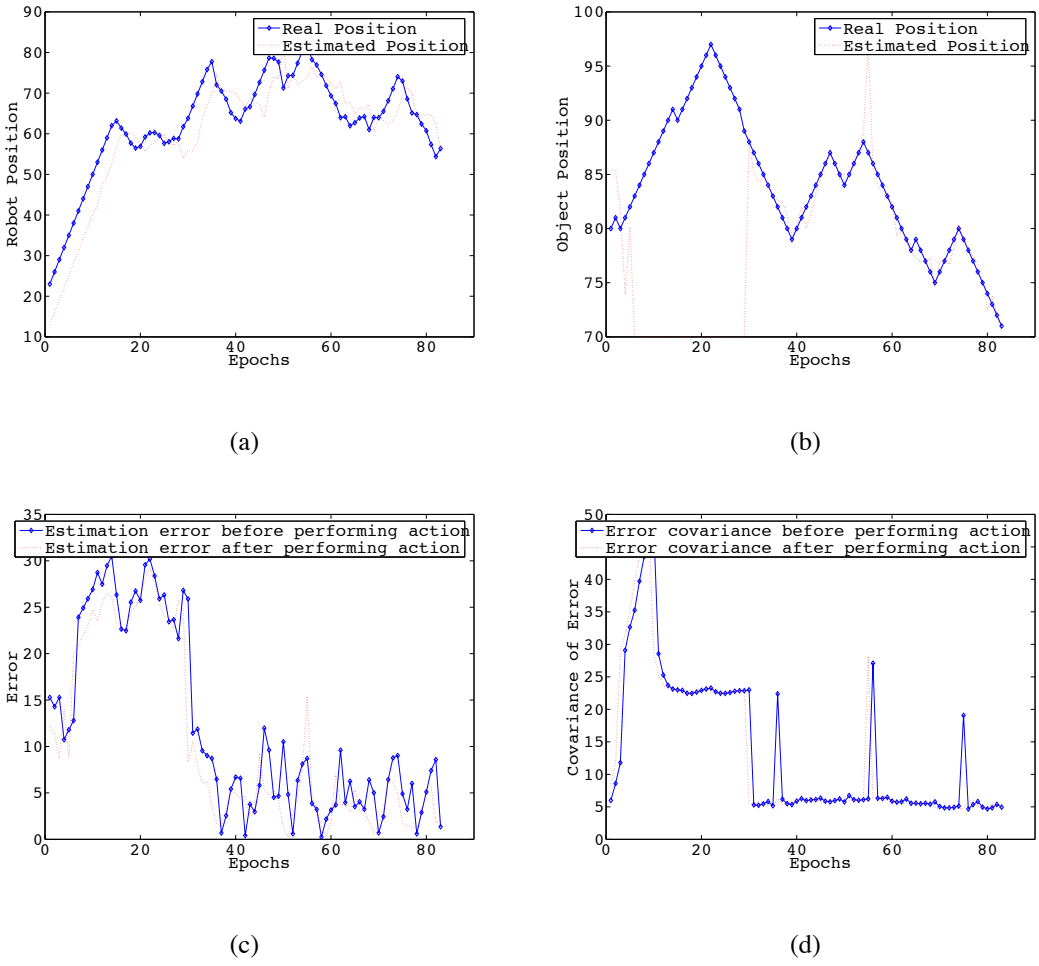


Figure 4.5: *The plots represent the results of the first simulation. The real and estimated positions of robot and object are compared in (a) and (b) respectively. Sum of squared error and error covariance before and after implementing actions are shown in (c) and (d) respectively.*

vation of tilt camera is less certain than the robot camera. Therefore we draw the sum of robot localization and object localization uncertainty in Fig 4.5(c). Although on the average the uncertainty is decreased, in some situations, the uncertainty after performing action is larger than the uncertainty before performing the action. The reason is that our action selection is based on the estimate. Nevertheless, the estimates are contaminated with noise and in reality, there is a difference between these two. This is why in some situations performing actions is not useful at all.

To see how the performance of an action influences the error, the variation of the error before and after implementing the action are drawn in Fig.4.5(d). Same as the uncertainty, the error is the squared sum of the robot and object localization error. As in the covariance case, we have a reduction in the average amount of error but in some cases the error increases after performing an action.

4.6.3 Simulation: Object Tracking Using a mobile Robot and Two Cameras

Here, we run a 2D simulation in which a robot and two pan-tilt cameras are trying to track the position of a target. Our goal is to estimate the position of the target with minimum possible uncertainty. There are three cameras: one is mounted on the robot and the others are mounted on the ceiling. The camera observations are uncertain.

We will use an optimization framework to find the best action.

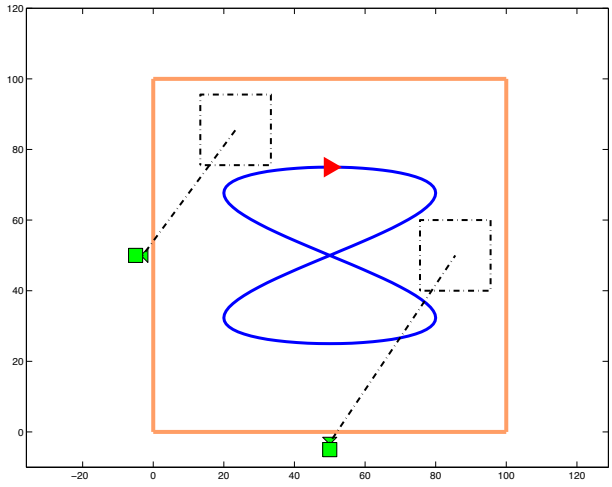


Figure 4.6: *In the simulation a square like environment is considered. A robot is trying to track a target using information of two ceiling mounted camera and its own camera. The environment is depicted in which the positions of the two cameras(green) and their field of view, the target path(blue) and the target initial position(red triangle) are specified*

4.6.3.1 The Goal

The goal of this simulation is to localize the robot and the object with minimum uncertainty. However, this can not be done unless the robot knows its own position with minimum uncertainty. So we have to consider these two targets at the same time. Considering the maneuvering skills of the robots and pan-tilt cameras, uncertainty can be adjusted by moving them to an appropriate position, as determined by solving the minimax cost problem. Actually the action should be taken in such a way so as to reduce the uncertainty of the estimation error.

4.6.3.2 Environment

The environment is divided to 100×100 squares. A schematic of the environment is drawn in Figure 4.6. In the environment there are two pan-tilt cameras, a robot and a target. The positions of cameras are considered such that they can cover the whole state space by changing pan and tilt angles.

4.6.3.3 The robot

Since the robot can get very close to the object, it can have the most certain estimation of the robot's position. Robot is equipped with an odometry sensor. The sensor is assumed to be unbiased and a Gaussian noise model is considered. The noise covariance is proportion to the distance traveled by the robot. To prevent having excessive error, the odometry sensor is periodically reset. The following mathematical model describes the sensors model.

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{4.51}$$

$$\sigma = \frac{|\, dis \,|}{10} \tag{4.52}$$

where dis is sum of absolute traveled distance and after every 6 steps, the odometry sensor is reset.

The robot is equipped with a camera. It is assumed that the observation of the camera is contaminated with a Gaussian noise. It is considered as unbiased sensor. The covariance of the Gaussian is proportional to the square of the relative distance. The range of robot vision in both directions is limited to 5 and 15 squares ahead.

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{4.53}$$

$$\sigma = \frac{dis^2}{100} \tag{4.54}$$

where dis is distance between the robot and the object

4.6.3.4 Pan-Tilt cameras

The main role of pan-tilt cameras is to guide the robot toward the object because the view range of robot's camera is much smaller than the pan-tilt. Two pan-tilt cameras are mounted on the two sides of the square. They can only observe a limited area but by changing the pan and/or tilt angles, they can cover the whole environment. They work independently and their angular pan and tilt speed are limited. We consider them as unbiased sensor with a Gaussian noise. The covariance of the noise is proportional to the relative distance between the object and the camera and also its distance from the optical axis. The same model is considered for the noise in both X and Y directions.

4.6.3.5 Object

The target follows a predefined path which is shown in Figure4.6. It is assumed that we have no information regarding the dynamics of the target. The only source of information about the position of the target are pan-tilt cameras and robots camera. When more than one source is available, the information is fused using WLC method.

4.6.3.6 Assumptions

As we mentioned before we assume that all the uncertainties have Gaussian form. It is also assumed that all sensors used in this simulation are unbiased. We also assume that after moving the object, there is enough time for the pan-tilt cameras and the robot to perform the optimal actions, i.e., the object does not move while the action is being executed.
We also assume that performing the action does not incur any cost.

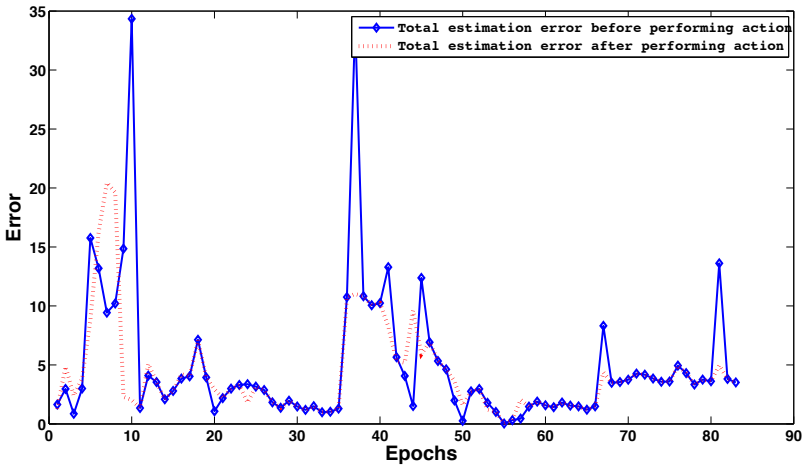


Figure 4.7: *The plots represents the results of the second simulation. Sum of error and error covariance before and after implementing actions are shown in (a) and (b) respectively.*

4.6.3.7 Constraints

Pan-tilt cameras are allowed to change their pan and tilt angles. However this change has been limited to some values. The maximum change in the pan and tilt angles for each time instant (angular velocity) is 5 degrees. The minimum and maximum pan angles for the camera are 0 and 60 and for the tilt angle is between -45 and 45 degree. Robot's speed is also limited. Robot displacement for each time instant (its velocity) is between -5 and 5 squares.

4.6.3.8 The Objective Function

The objective function is a function that quantifies the benefits and costs of different action sets. Here we did not consider any cost for performing the actions. We only consider the cost related to uncertainty of the estimation. Since all the uncertainties have Gaussian

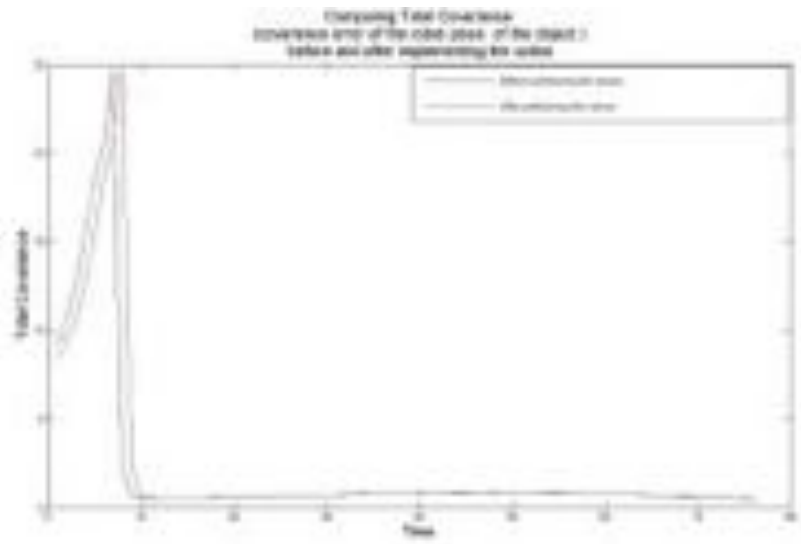


Figure 4.8: *The plots represents the results of the second simulation. Sum of error before and after implementing actions are shown in (a) and (b) respectively.*

forms, it is logical to consider the covariance matrix of the Gaussian belief to represent the amount of the uncertainty involved in the belief. The cost function has a quadratic form and we can solve the convex optimization problem.

4.6.3.9 Initial Beliefs

We assume that, initially, we have information regarding the positions of the robot, and the object(target). Also, we are aware of pan-tilt angles. These are in the form of Gaussian distribution.

4.6.3.10 Results

The simulation is ran for 1000 times of a pre-specified target path. KF is used to estimate the position of the robot. Since we have assumed there is no information regarding the

target dynamic, we use WLC to fuse and estimate the positions of the target if more than one camera observe the target. Figure 4.7 shows mean estimation of sum of robot and target localization error. As we can see from the figure, the implementation of the action reduces the error in most of the cases. However, in some cases, we have an increase in the amount of error which is due to the presence of estimation errors. During the early stages, where the localization error is high, implementing the action is more effective. We can also conclude that it is not necessary to execute an action in each step, especially when the error is lower than a threshold. In Figure 4.8, the mean of the sum of error covariance before and after implementing an action are drawn. At first the difference between these two are clear. Later this difference vanishes and become negligible. The main reason is the robot is close enough to the object and changing its position does not have a big influence on the covariance. However, on average we have a reduction in the amount of uncertainty.

4.6.4 Simulation: Object Tracking Using a Mobile Robot and Camera with Action Cost

As we explained before, in the previous simulations, we did not consider any cost for performing the actions. As a result, in every step of the simulation we performed actions which in some cases were not useful, e.g., the error covariance did not decrease. Here, we consider cost for implementing the actions. With this change, we ran the 1D simulation. We only performed the action if and only if the benefit of the action was more than its cost. Comparing the results of this simulation, shown in Fig. 4.9, with the first simulation results, the robot and tilt camera only performed actions in 50% of the cases. In other words, if a similar cost is considered for all the actions, in the second simulation, we only need to

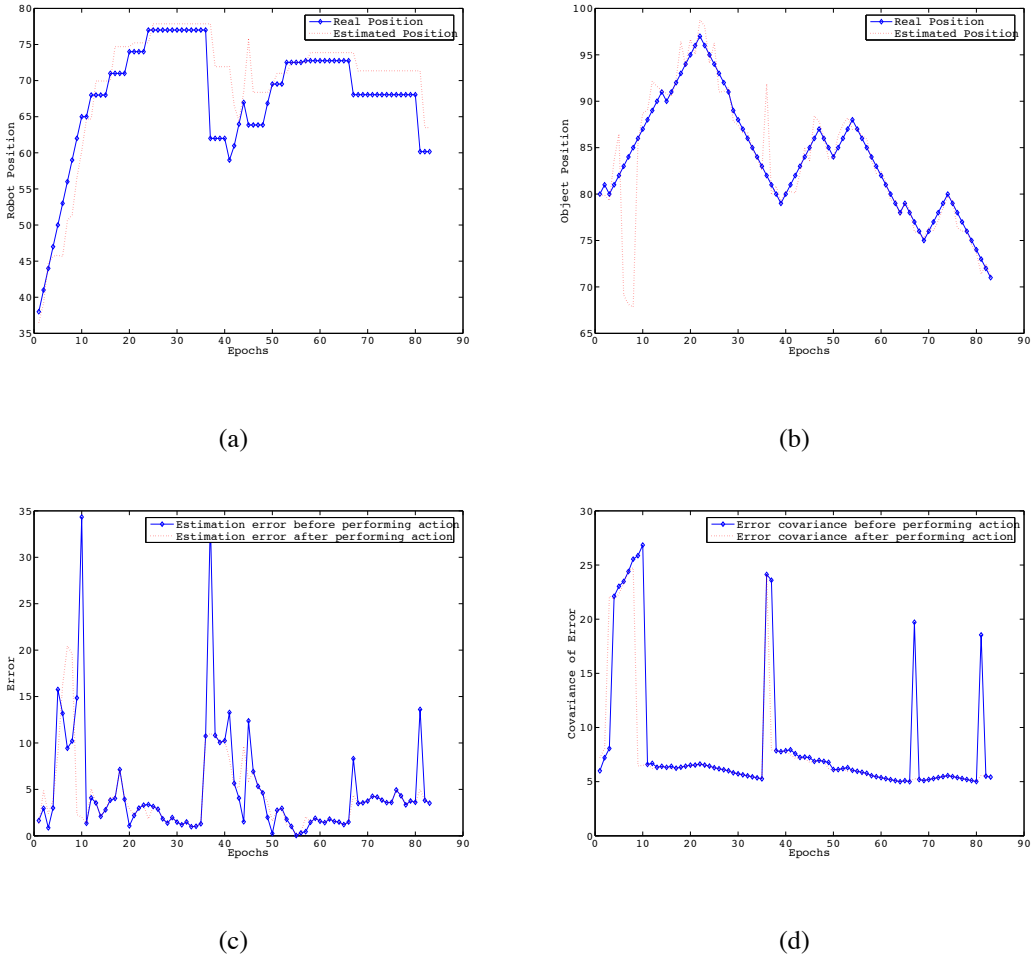


Figure 4.9: *The plots represents the results of the third simulation. In this simulation we consider cost for implementing actions. The real and estimated positions of robot and object are compared in (a) and (b) respectively. Sum of error and error covariance before and after implementing actions are shown in (c) and (d) respectively.*

spend half of the cost comparing to the first simulation case. This is due to the shallow form of the sensors model in its range. In other words, a small change in relative position of the robot and object can not result in huge reduction in the error covariance.

Chapter 5

ACP and Cooperative Path Planning

5.1 Introduction

We applied ACP to one particular problem, namely how to plan paths for robots taking into account the coverage of the camera network as well as the robots' own sensors.
In many NRS, surveillance cameras will run a set of event detection algorithms, for instance observing events such as people waving or other emergencies, each with a different priority. Therefore, after detecting the events, if needed, robot/s should be deployed to the position. However, there are two major questions:

- The first question is : which robots should go to which goal and in which order (what if a robot has to serve several goals)?
- The second question is related to the path each robot should take to reach the goal. Which path should the robot follow?

In general, there is a large number of possible paths. An example of such a scenario is shown in Fig. 5.1. The figure represents the map of the URUS test bed which is located in

UPC Nord Campus, Barcelona, and includes several networked mobile robots and surveillance cameras. Ubiquitous Networking Robotics in Urban Settings (URUS) is a European project aimed to develop new technologies for establishing cooperation between network of robots and human beings and/or the environment in urban areas. Active Cooperative Perception is a part of the project. This project is jointly designed and implemented by a team of researcher from 13 different universities, research centers and companies. As we can see from the figure, there are two robots (named as R1 and R2) located at (-5,100) and (90,45) respectively) that should serve two goals (G1 and G2 located at the positions (35,65) and (40,78) respectively). In the figure, some of the possible paths for the robots are drawn. If the path length is the only parameter of interest, R1 should go to G2 and R2 to G1. If we assume the size of the R2 robot is larger than R1, serving G1 by the R2 robot may cause the robot to hit the wall while passing the corridor in front of B6 building or the objects around while is maneuvering in the area close to FIB Square to avoid the obstacles. If the possibility of bumping into a wall or obstacle is considered as a criterion of path quality then the R2 robot should go to G2 and R1 to G1, although these paths are longer compared to the previous ones. Another issue concerns which path R1 should take if it had to go to goal G2: the path passing through the corridor in front of C5 building (which is wider but is not covered by the camera network) or behind it (which is covered by camera C1 (placed at (100,100))? The robot takes the second path if the most informative path (in the sense of keeping the robot it well localized using cameras information) should be taken. Therefore, we are looking for an algorithm that provides an optimal and real time solution for multi-robot multi-goal path planning. Another issue is that the algorithm must be sensitive to changes in the environment and react in real time, i.e., it should be active. For instance, if one of the goal moves, or an urgent inquiry is received, the system has to react

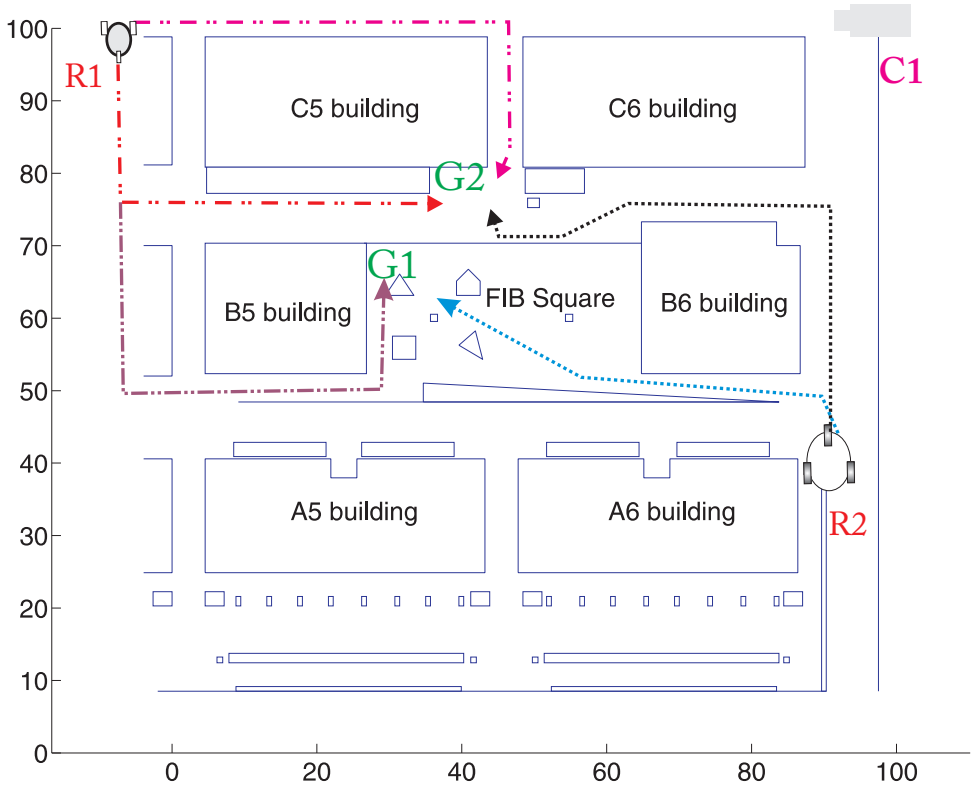


Figure 5.1: *The figure shows a schematic diagram of URUS site. The central station receives two requests and two robots (R1 and R2) are ready to be deployed. Depend on the parameters of interest, there are a number of different paths and plans. Some possible paths are drawn in the figure.*

and re-plan in real time. As we mentioned before, all information is sent to central station (CS) and it issues the command. However, considering the CS traffic and bandwidth limitation, assuming that the robots receive information about the latest environmental change, they should be able to decide independently and uniformly. This means the solution should be applicable to either a centralized or decentralized architecture. Briefly, the main idea of this paper is to define appropriate criteria and find a way to actively and optimally plan paths for a team of robots in real time in either a centralized or decentralized manner.

Markov Decision Process (MDP) framework is used to address our sensor-aware path planning problem [85, 12]. A decision-theoretic framework such as a MDP is a natural way to express concurrent and possibly conflicting objectives such as reaching the goal quickly, keeping the robot localized, keeping the target in sight, each with their own priority. Given the partially observable nature of the problem, modeling it as a partially observable MDP (POMDP) would be appropriate. However, given the scale and level of detail of the problems we are targeting, with many states, and, more importantly, a large number of possible observations and a high planning horizon, this is beyond current state-of-the-art of tractable algorithms to solve POMDPs. Therefore we assume full observation.

Here, we first show how to find the optimal plan when we only have one robot and one target. Then, we explain how to extend the idea for a multi-robot multi-goal case.

5.1.1 Background on Markov Decision Processes

We will briefly introduce the Markov Decision Process (MDP) framework [85, 12]. MDPs provide strong mathematical tools for decision-making under uncertainty, in case the state

Initialize V arbitrarily, e.g., V (s) = 0, $\forall s \in S$ to zero

repeat

$\delta \leftarrow 0$

for all $s \in S$ **do**

$v \leftarrow V(s)$

$\mathrm{V(s)} \leftarrow max_{a \in A} \left\{ R(s,a) + \gamma \sum_{s' \in S} p(s|a,s') V(s') \right\}$

$\sigma \leftarrow max(\sigma, |v - V(s)|)$

end for

Until $\delta < \varepsilon$

Return V

Table 5.1: *Value Iteration Algorithm (from [85]).*

of the environment is observable to the robot. A MDP is formally specified by a four tuple (S, A, T, R) where S is a (finite) set of states, A is a (finite) set of atomic actions, T is the transition model and R is a reward function. Each element of S describes the state of the system at a given time instant. A policy is a function $\pi : S \rightarrow A$ which maps states to actions. $\pi(s)$ states the action that should be taken in state s. The value of policy π when the system starts at state s and continues for an infinite horizon is defined and determined as:

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right] \tag{5.1}$$

where $0 \leq \gamma \leq 1$ is a discount factor. A value function is a mapping $V : S \rightarrow \mathbb{R}$ that determines the sum of total expected future reward when starting at state s. The value of the optimal policy V^* that tells us which action to take at each state in order to maximize the total expected reward can be determined recursively as:

$$V^*(s) = \max_{\pi} V^{\pi}(s) = \max_{a \in A}\left\{R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')\right\}. \tag{5.2}$$

In order to compute V^*, dynamic programing techniques such as value iteration can be used [85, 12].

5.2 Active Cooperative Path Planning

Here, we address the problem of finding an optimal path for a robot considering different criteria [61]. In order to determine the optimal path, first, we have to define these criteria to evaluate the path. These criteria can be translated into rewards, e.g., if the criterion of interest is the path length, when we take a shorter path, we receive a larger reward. When we consider several different criteria we should combine them to get a total reward. We

will define the total reward as a linear combination of the individual rewards. Then, we formalize the problem and associate an objective function to the total reward. Finally, we show similarities between MDP problem and our problem and use the MDP solution to solve our problem.

5.2.1 Cost and Rewards for Planning a Path

We will implement our ideas based on decision-theoretic robot guidance by defining the MDP's reward function. This is a flexible way for the user of the system to specify the relative importance of the considered factors. In particular, the idea of taking the best path is directly related to costs and rewards. By rewards, we mean what the agents receive along the path or at destination. The costs are defined as the amount of resource consumption, effort, loss necessary to achieve the goal or the risk, e.g., risk of bumping into an obstacle due to taking a narrow path. In our scenario, localization certainty, visibility of the target location, as well as reaching the destination are considered as the rewards. Maneuvering risk and traveling are considered as costs, i.e., as negative rewards. Each of them are explained below in detail.

Before going into details, it is necessary to mention that the world is discretized in a number of states. Each state is specified by its position and its orientation. The orientation space is divided into eight equal sectors, the first starts at zero and each step is $\frac{\pi}{4}$. There are three atomic actions possible in each state: stay in the same state but change the orientation $\pm\frac{\pi}{4}$, or move to one of the 8 adjacent states without changing the orientation.

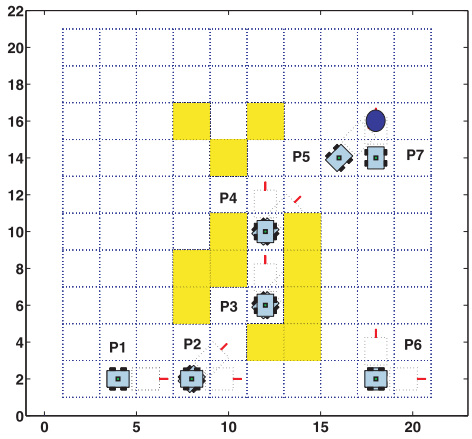


Figure 5.2: *Illustration of the maneuvering cost function. Giving more attention to maneuvering cost than the traveling cost, in absence of other costs and rewards, forces the robot to take path P1-P2-P6-P7 instead of P1-P2-P3-P4-P5, although P1-P2-P3-P4-P5 is shorter.*

5.2.2 Maneuvering Cost

Often, a robot needs to change its orientation. To do so, it needs space. In larger spaces, the maneuvering risk is smaller. For a robot, it is less possible to bump into an obstacle when it has a larger free space to maneuver. The places closer to the obstacle are more risky for changing the orientation. Moreover, a narrow passage is more risky to take than a wider passage. Therefore, the maneuvering cost ρ_{M} of each state is defined as a function of two factors:

$$\rho_{\mathrm{M}} = \frac{\alpha_M}{\lambda} + (1 - \alpha_M) * \vartheta \tag{5.3}$$

where $0 \leq \ \alpha_M \leq 1$ is used to balance the importance of passage width (first term) versus the number of surrounding obstacles (second term) and ϑ is the number of surrounding obstacles. λ is defined as:

$$\lambda = \frac{\lambda_i}{\lambda_{max}} \tag{5.4}$$

λ_i is the cell size of i^{th}cell and λ_{max} is the largest cell size.

To make things clear, a scenario is explained in Fig. 5.2. Considering a higher cost for maneuvering, the robot is forced to take path P1-P2-P6-P7 instead of taking the path P1-P2-P3-P4-P5, even though the second path is shorter.

5.2.3 Traveling Cost

The cost of traveling ρ_{T} has two components: the relative distance and rotation. It is considered as a linear combination of the two costs:

$$\rho_{\mathrm{T}} = \alpha_T * \Delta_p + (1 - \alpha_T) * \Delta_\theta, \tag{5.5}$$

where $0 \leq \alpha_T \leq 1$.

It includes two components. The first component is calculated based on the Euclidean distance Δ_p the robot needs to take to travel from one state to another. The second component is determined by calculating the absolute difference Δ_θ between the orientation of the two states. We usually give the higher relative importance to the second term as for our robots changing the orientation needs more resources in terms of energy and time. Moreover, for changing the orientation, the robot has to stop and then change its orientation, which takes time. Increasing the localization error during the orientation change due to slippage is another reason to assign higher cost to orientation change.

5.2.4 Goal Reward

The goal reward ρ_G is defined as the reward the agent receives when it reaches the goal state. This reward may vary based on the degree of our interest in the goal and the situation.

For example, if the camera network detects a fire and the system deploys the robot to provide more details, considering the urgency of the case, the system only considers the goal reward which result in generating the fastest path to the goal and ignores other possible rewards.

5.2.5 Visibility Reward

One important issue in our scenario is the visibility issue. The visibility is defined as feasibility of observing the object of interest at a specific position and angle. We explain this concept by providing an example which is drawn in Fig. 5.3(a). In this example, a robot with an on-board camera in several positions is shown. The robot in P1 is not able to see the object of interest which is depicted by a circle because its line of sight is blocked by the obstacle on the way. However, in P2, the robot is potentially able to view the object. It means that although the object may not be in robot camera view field, there is no obstacle that blocks the robot line of sight. The robot in P3 can see the target and we give a higher visibility reward compared to P2 because as it is closer to the object and the visibility is less sensitive to change in the orientation. Moreover, in P4, a zero visibility reward is assigned. Although it is closer to the object than the robot in P2, considering its orientation, the target is not in the robot's line of sight.

One issue in regard to visibility is timing priority and distance priority. Sometimes, a robot should quickly reach a point where the object of interest is visible. For example, a robot with a long range sensor, e.g., a laser, should quickly reach a point where it could detect an intruder. In such a scenario, time has the priority. It does not need to really get close to the object; even from a far distance, it can discover the existence of the object. On the

other hand, in an object recognition scenario, where we should closely approach the object in order to recognize it, e.g., a face detection scenario, time is not a priority any more, but rather the distance. Formally, ρ_V is defined as:

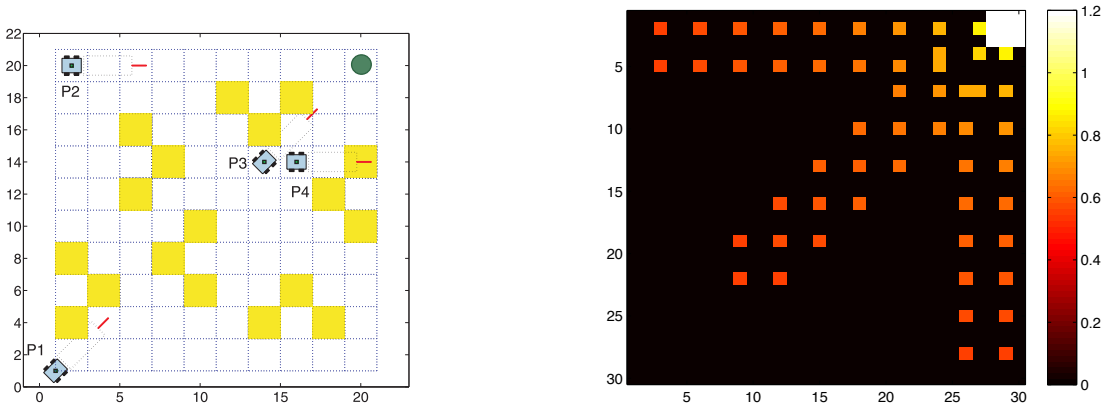
$$\rho_V = \alpha_t \alpha_{vi} \Delta_p + \alpha_d \frac{\alpha_{vi}}{\Delta_p} \tag{5.6}$$

where α_t is 1 if time has the priority, zero otherwise, and α_d is 1 when distance has the priority, zero, otherwise. Δ_p is defined as the Euclidean distance between the goal and the state and η is a positive number representing the maximum visibility radius. Δ_θ is the relative angle between the robot's orientation and the line of the sight to the goal and ξ represents the maximum visibility angle. $\alpha_{vi} = 0$, If $\Delta_p > \eta$, $|\Delta_\theta| > \xi$, line of the sight is blocked by an obstacle placed between the state and the goal or an obstacle is in the state; Otherwise $\alpha_{vi} = 1$. The visibility and the robot sensor range are related but visibility is a different concept, as it is affected by the robot orientation and, more importantly, the path characteristics. A path with many obstacles between the goal and the robot has a low visibility, even if the robot is equipped with a long-range sensor.

The visibility reward ρ_V for the goal state positioned in Fig. 5.3(a) is drawn in Fig. 5.3(b)($\alpha_d = 1, \alpha_t = 0$). In Fig. 5.3(b), we do not consider any obstacle to show the distribution of the visibility reward. The color intensity is proportional to the visibility reward value, the higher the reward, the lighter the square face color.

5.2.6 Localization Certainty Reward

Often, the pose of a target, e.g. a robot, a person, etc. is an important piece of information we need to know. For example, when the robot should approach a person to let the person



(a) (b)

Figure 5.3: *The left figure shows a robot in several positions. The reward in Positions P1 and P4 is zero but in P2 and P3 is not zero. The reward value depends on the relative distance and the angle receives. The right figure shows the visibility reward ρ_V for the goal state at the top right corner. We do not consider any obstacle to show the distribution of the visibility reward.*

to interact with the robot, in order to prevent collision, having an accurate relative localization of robot and the person is very important. In other words, if the person is localized but with a large uncertainty, the robot can use its sensors in order to help the camera network to better localize the person. Therefore, if the localization uncertainty of the robot is not good enough, while it is traveling toward the person, it has to give more priority to paths with larger *Certainty Reward* than to other paths, e.g., a shorter path but with a large localization uncertainty. The observation model of the surveillance cameras is assumed Gaussian, with the mean centered at the real value. The covariance proportionally increases with the relative distance between the camera and object of interest[36]. To each state in state space we assign a real number ρ_{L} which is called Localization Certainty and is defined as:

$$\rho_{\mathrm{L}} = \frac{1}{1+\sigma_i} \tag{5.7}$$

where σ_i is defined as:

$$\sigma_i = \frac{1}{e^T \Sigma^{-1} e} \tag{5.8}$$

where $\mathbf{e} = [1,1,...,1]^T$ is a N-dimentional vector, N is the number of cameras that can observe the state and Σ is the covariance matrix of the observation model of the cameras which cover the state.

5.2.7 The Objective Function

The total reward is defined as linear combination of the parameters defined in 5.2.1:

$$\rho = \beta_G \rho_G + \beta_V \rho_V + \beta_L \rho_L + \beta_T \rho_T + \beta_M \rho_M \tag{5.9}$$

To make it easy to compare, all rewards (ρ's) are normalized ($\rho \in [0,1]$). Although, there is no limitation and the total reward can be defined as a non-linear function of the rewards,

a linear function is preferred because prioritizing and adjusting the rewards (by changing β's) is easier. Choosing a larger β for a reward, causes the total reward is to be affected more by that reward. Setting β to zero removes the effect of the reward.

After defining the total reward, we can define our objective function. We make it a function of the total reward received when the robot implements an action a_t and moves from pose s_t to pose s_{t+1} where t is time tag and s and a represent robot's pose and action respectively. It is defined as the expected rewards received over an infinite horizon.

$$J = E\left[(\sum_{t=0}^{\infty} \delta^t \rho_{s_t,a_t,s_{t+1}})\right] \tag{5.10}$$

where $0 \leq \delta < 1$. Our intention is to maximize the objective function or the reward received over an infinite horizon:

$$\max_{a_t} J = \max_{a_t} E\left[(\sum_{t=0}^{\infty} \delta^t \rho_{s_t,a_t,s_{t+1}})\right] \tag{5.11}$$

Comparing the two equations (5.10) and (5.1), we can find a similarity between our problem and a MDP. In both equations the idea is to optimize a reward function for an infinite horizon. Position of a robot can be considered as its state and changing the robot's state is similar to state transition in MDP. If the MDP assumptions are valid, we can use the MDP to address our problem.

5.2.8 Experiments

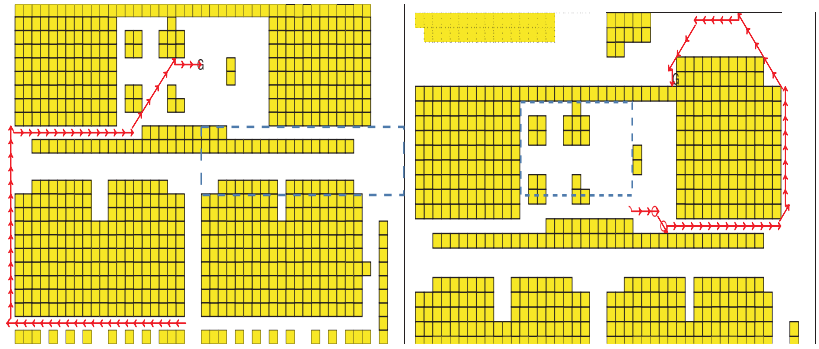
To verify the performance of the proposed method, we ran a series of simulations. Fig. 5.1 shows the schematic diagram of the URUS test bed which is located in UPC Nord campus,

Barcelona. The area size is about 1 hectare, which we divided in equal size $2 \times 2\, m^2$ squares. In each cell, we considered 8 different robot orientations. The first orientation is at 0 radian and the step is $\frac{\pi}{4}$. The total number of states is 20000. However, part of them are occupied by obstacles and we only deal with the free states. Fig.5.4 presents the free spaces and obstacles. The places marked with yellow filled squares are the obstacles and the rest is the free space. A discrete MDP is used to model the path generation. The reward function is considered according to (5.9). The basic atomic actions are either to stay in the same cell and only change the orientations $\pm\frac{\pi}{4}$ or move forward. Here we assume deterministic actions, however it is trivial to extend the work to noisy actions, as the same value iteration procedure can be applied. In Fig. 5.4, the goal position is specified by 'G' and the area under the camera coverage is shown by a rectangle with a dashed edge.

Consider the case when the network of cameras detects a fire. The robot should be deployed in such a way it gets to the place in the shortest possible time. Another situation is where the robot is asked to approach and provide a service for a person who is localized but with large uncertainty. To do so, robot has to know its own localization very well in order to find out the position of the person using the relative localization for further operation. This is the situation where robot has to take a path under camera coverage and with acceptable ρ_L. The aim of the experiments are to evaluate the effect of different parameters on the generated path. Fig. 5.4(a) shows a scenario where we have a camera which covers the area marked with the dashed rectangle. We check the behavior of the system, the generated optimal path, by changing the values of β_L and β_T while β_V is set to zero. First, we set β_L to zero. Naturally, the generated path is the path with the lowest traveling cost. In Fig. 5.4(a), the generated path is shown. In Fig. 5.4(c), we kept all β's the same but change β_L. Increasing β_L causes a different path to be considered for the robot. The generated path

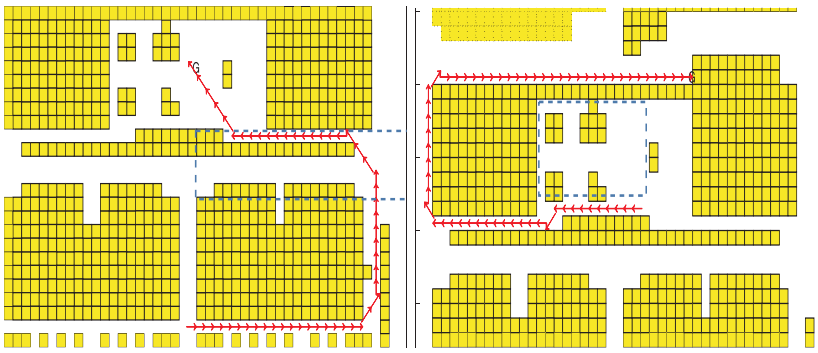
goes through the area covered by the camera. To see the further effect of increasing β_L, we use the same setting but increase β_L. This time the system changes the generated path in such a way it stays longer under the area covered by the camera. The result is shown in Fig. 5.4(e). It can be seen that even when we change the robot orientation, due to a large β_L, the system still guides the robot to the area covered by the camera.

The next scenario is designed to see the effect of β_V and β_T on the generated path while either β_L is fixed or changed. There are situations where sending the robot to the position where the object is in robot line of sight has the top priority e.g., the camera detects an intruder and has to send the robot to track. In other words, the priority is that the robot reaches to a point in which it can observe the person as quickly as possible. Using a robot equipped with a laser range finder, the system can then track the person. Fig. 5.4(b) shows the case where we set all β_V and β_L to zero. This is similar to Fig. 5.4(a) but the starting and goal locations are different. The generated path has the lowest traveling cost. In the second case we gave more priority to β_V over β_T. As we can see in Fig. 5.4(d), the robot is provided with a different path. Only considering the traveling cost, this path is more costly. However, taking this path, causes the robot to reach earlier a point at which the object is in its line of sight. We have an area under camera coverage close to starting state. For improving robot localization uncertainty, giving some weight to β_L, causes the path generated to become longer but pass through the area covered by the camera. This is depicted in Fig. 5.4(f).



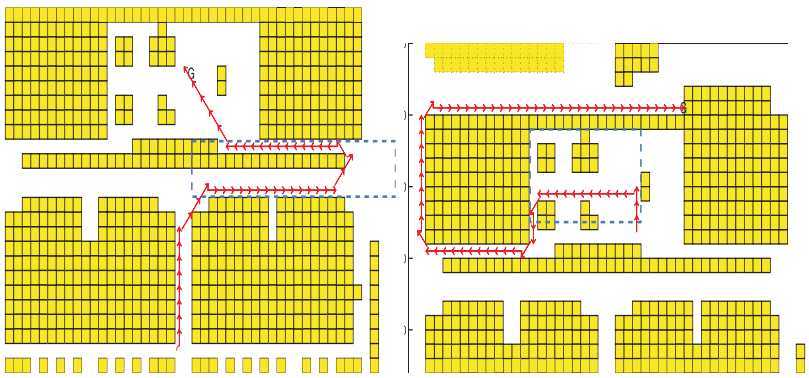
(a) $\{\beta_G = 100, \beta_V = 0, \beta_L = 0, \beta_T = -1, \beta_M = 0\}$

(b) $\{\beta_G = 100, \beta_V = 0, \beta_L = 0, \beta_T = -1, \beta_M = 0\}$



(c) $\{\beta_G = 100, \beta_V = 0, \beta_L = 100, \beta_T = -1, \beta_M = 0\}$

(d) $\{\beta_G = 100, \beta_V = 100, \beta_L = 0, \beta_T = -1, \beta_M = 0\}$



(e) $\{\beta_G = 100, \beta_V = 0, \beta_L = 300, \beta_T = -1, \beta_M = 0\}$

(f) $\{\beta_G = 100, \beta_V = 100, \beta_L = 100, \beta_T = -1, \beta_M = 0\}$

Figure 5.4: *The figures represents the generated path in different situations. In the figures, the dashed rectangle corresponds to a region covered by cameras, while the yellow squares are obstacles. To make the results more visible, only part of the environment shown in 5.1 in which the scenarios take place are shown.*

5.3 Task Assignment For a Team of Robots

In Section5.2, the problem of robot guidance for a single goal and a single robot is addressed. However, often in URUS scenario, the CS receives several requests at the same time and it has to deploy a number of robots to serve them. An example of such a scenario is illustrated in Fig. 5.6(a). As shown in the figure, the CS recognizes three requests (1, 2 and 3) and there are two robots (A and B) available. Three possible different solutions (among others) are drawn in Figs. 5.6(b), 5.6(c) and 5.6(d). Each solution is called a plan. A plan s_k is a vector of length $M \times N$ where M is the number of robots and N is the number of goals. s_k can be written as: $s_k = [e_{11} \cdots e_{1N}\ e_{21} \cdots e_{2N} \cdots e_{M1} \cdots e_{MN}]$ where each vector element e_{ij} is either 1 or 0. It is 1 if the goal j is served in the plan by robot i, 0 otherwise. For example, the plan shown in Fig. 5.6(a) in which none of the goals are served can be represented by $s_1 = [0\ 0\ 0\ 0\ 0\ 0]$. Fig. 5.6(b) represents another plan. The plan can be expressed as: $s_2 = [1\ 0\ 0\ 0\ 1\ 1]$. It says that robot A serves goal 1 and robot B serves goals 2 and 3. Each plan tells us about the resource allocation and the plan sequence tells us about the serving order. According to the plan shown in Fig. 5.6(b), robot A only serves demand 1. Robot B, serves demand 2 and 3. Another solution can be to serve demand 3 by robot A and demands 1 and 2 by robot B respectively. This plan is sketched in Fig. 5.6(c). We also could keep robot A in standby mode (in order to serve the possible urgent demands) and serve all demands by robot B. This plan is shown in Fig. 5.6(d). Actually, the order can be concluded by knowing the plan and prior plans, the plan that should be implemented prior to the plan.

Now, let us assume we have a set of plans. The question is which plan is optimal. Therefore, we need a way to evaluate them and determine the best one. To do so, a set of

criteria is defined. These criteria are a way to evaluate a plan. For instance, if the execution time of a plan is the criterion of interest, the plan with the smallest execution time should be chosen. A reward is defined proportional to each parameter, e.g., the plan with the smallest execution time receives more reward (less cost) than others. However, in order to find the optimal plan, we need to consider the plans that should be executed prior to the plan as well. When we have more than a reward, a collective reward is defined. Based on the collective reward, an objective function should be defined. Finally, optimizing the objective function gives us the optimal plan.

5.3.1 Costs and Rewards for Optimal Plan

The definition of the final objective function is based on the rewards we define. In our case, we consider goal dissatisfaction cost ρ_{DS}, time expectancy reward ρ_{TE} and resource availability ρ_{RA}. Below, we explain them in details.

5.3.1.1 Goal Satisfaction Reward

We consider a reward for serving each individual goal. The amount of the reward depends on the degree of goal importance. The more important the goal the more reward the robot receives. For a person desperately asking for a service, e.g., asking fr help to be taken to a hospital, a high reward is assigned. On the other hand, a lower reward is assigned to a request of taking a tour around. In general, the goal satisfaction reward ρ_{gs_s} for a plan s is defined as:

$$\rho_{GS_s} = \sum_{i=1}^{N} \alpha_{GSs_i} \rho_{GS_i} \tag{5.12}$$

where $\alpha_{GS_{\mathsf{s}_i}}$ is 1 when i^{th} goal is served in plan s, otherwise, zero. ρ_{GS_i} is a positive number and represents the reward received for serving i^{th} goal and N is the number of goals.

5.3.1.2 Goal Dissatisfaction Cost

The dissatisfaction cost is defined as the punishment received for not serving a goal. It means that not only no satisfaction reward is received but also we get punished for it. For example, if an ordinary goal (e.g., taking a tour) is not served in a plan, it does not receive a satisfaction reward and it might not be punished. However, if an urgent inquiry (e.g., transferring to a hospital) in a plan is denied, not only no satisfaction reward is received but also the plan can be punished for that. Similar to goal satisfaction reward, goal dissatisfaction cost for a plan s is defined as:

$$\rho_{DS_{\mathsf{s}}} = \sum_{i=1}^{N} \alpha_{DS_i} \rho_{DS_i} \tag{5.13}$$

where α_{DS_i} is 0 when the goal is served, otherwise, 1. N is the number of goals and ρ_{DS_i} is the cost received for not serving i^{th} goal. It is a negative number, the smaller the number means the more punishment for not serving a goal.

5.3.1.3 Time Expectancy Cost

In a team of robots, often, the tasks are shared among the team members. Therefore, in plan s_k of each robot i, an execution time $\tau_{\mathsf{s}_{k_i}}$ is related. It is a vector of length N and defined as: $\tau_{\mathsf{s}_{k_i}} = \left[\tau_{\mathsf{s}_{k_{i1}}} \cdots \tau_{\mathsf{s}_{k_{iN}}} \right]$ where $\tau_{\mathsf{s}_{k_{ij}}}$ is the time needed by robot i to reach the goal j. $\tau_{\mathsf{s}_{k_{ij}}}$ is a positive number ($\tau_{\mathsf{s}_{k_{ij}}} > 0$) and its value depends on robot velocity, robot to goal relative distance and the path the robot takes. It is defined as:

$$\tau_{\mathsf{s}_{k_i}} = \sum_{j=1}^{N} \alpha_{\mathsf{s}_{k_{ij}}} \tau_{\mathsf{s}_{k_{ij}}} \tag{5.14}$$

$\alpha_{s_{k_{ij}}} = 1$ if goal j is served in plan s_{k_i}, otherwise, zero. τ_{s_k} is a vector length MN and is defined as $\tau_{s_k} = [\,^{\tau_{s_{k_1}}} \cdots ^{\tau_{s_{k_M}}}]$. Time Expectancy Cost of plan s_k is defined as:

$$\rho_{\tau_{s_k}} = \max(\tau_{s_k}) = \max(\tau_{s_{k_1}}, ..., \tau_{s_{k_M}}) \tag{5.15}$$

5.3.1.4 Resource Unavailability Cost

Robots as our resources are the providers of the services. However, the number of robots are limited. When we receive requests, one option is to deploy them all. There are time instants when we are expecting an urgent inquiry and want to keep some of the resources, the robots, in standby mode to serve the possible upcoming requests. Therefore, if M_r and $M_{a_{s_k}}$ represent the number of robots requested to be kept as standby and number of available robots in plan s_k respectively, then:

$$\rho_{RA_{s_k}} = \begin{cases} 0 & M_r \leq M_{a_{s_k}} \\ (M_{a_{s_k}} - M_r)\alpha_{RA_i} & M_r > M_{a_{s_k}} \end{cases} \tag{5.16}$$

where α_{RA_i} is a positive number and it value represents the punishment considered for unavailability of a robot.

5.3.2 The Total Reward

In Section 5.3.1, a number of costs and rewards are explained and formalized. In general, the total reward ρ_i that i^{th} robot receives can be defined as a function of individual rewards $\rho_1, ..., \rho_W$:

$$\rho_i = f(\rho_1, ..., \rho_W) \tag{5.17}$$

Here, we consider f as a weighted linear combination of the the individual rewards:

$$\rho_i = \gamma_{i1}\rho_1 + ... + \gamma_{iW}\rho_W \tag{5.18}$$

where w represents the total number of robots, γ_{iw}'s are called the reward coefficients of i^{th} robot, $0 \leq \gamma_{iw} \leq 1$. By adjusting the reward coefficients, we can prioritize one type of rewards over the others and therefore the feedback received from the environment can be changed. Therefore, change in configuration space and our needs is translated into the change in the reward's coefficients.

The total reward received for a team of robots as a result of implementing action a_t and moving from plan s_{k1} to s_{k2} is defined as:

$$\mathsf{R}_{\mathsf{s}_{k1},\mathsf{a}_t,\mathsf{s}_{k2}} = \sum_{i=0}^{M} \rho_i \tag{5.19}$$

where a_t is a vector of length M and i^{th} element of the vector represents the action performed by robot i.

5.3.3 The Objective Function

To evaluate different plans, we define an objective function based on the total reward a plan receives:

$$\mathsf{J} = f(\mathsf{R}_{\mathsf{s}_{k1},\mathsf{a}_t,\mathsf{s}_{k2}}) \tag{5.20}$$

Here, the objective function is considered as expected discounted sum of rewards received over an infinite horizon:

$$\mathsf{J} = E(\sum_{t=0}^{\infty} \delta^t \mathsf{R}_{\mathsf{s}_{k1},\mathsf{a}_t,\mathsf{s}_{k2}}) \tag{5.21}$$

where $0 \leq \delta < 1$. Our intention is to optimize the objective function:

$$\mathsf{J}^* = \max_{\mathsf{a}_t} E(\sum_{t=0}^{\infty} \delta^t \mathsf{R}_{\mathsf{s}_{k1},\mathsf{a}_t,\mathsf{s}_{k2}}) \tag{5.22}$$

After optimizing, we get a sequence of actions. This action sequence which is named a policy is a mapping from plan space to action space, in each plan we should know the optimal action.

5.3.4 MDP and Optimal Plan

Let us assume there are K possible plans: $\mathsf{S} = \{\mathsf{s}_1, \mathsf{s}_2, ..., \mathsf{s}_K\}$. Each plan is a vector of length MN. We also define transition function T as:

$$\mathsf{T} : \mathsf{S} \times \mathsf{S} \rightarrow [0, 1] \tag{5.23}$$

The transition function tells us the probability of moving from a plan to another one. Fig. 5.5 shows an example of a scenario in which two robots (A and B) should serve three goals(1, 2 and 3). In the Fig. 5.5 four plans $\mathsf{s}_1 - \mathsf{s}_4$ are shown. The top left end right are called plans s_1 and s_2 respectively. If the current plan is s_1, by implementing action a_1 we can move to plan s_2 with probability 1. On the other hand, it is not possible to move from s_2 to s_1, $\mathsf{T}(\mathsf{s}_2, \mathsf{s}_1) = 0$.

Let's consider $\mathsf{A}_{\mathsf{s}_i}$ as the set of all possible actions in plan s_i. For example, if the current plan is s_1 (Fig. 5.5), there are only two possible actions: by implementing action a_1 we move to plan s_2 (the top right plan in the figure) and by implementing action a_2 we move to plan s_3 (the bottom left plan).

Let's consider $\mathsf{R}_{\mathsf{s}_i,\mathsf{a}_k,\mathsf{s}_j}$ as the reward received as a result of implementing action a_k and

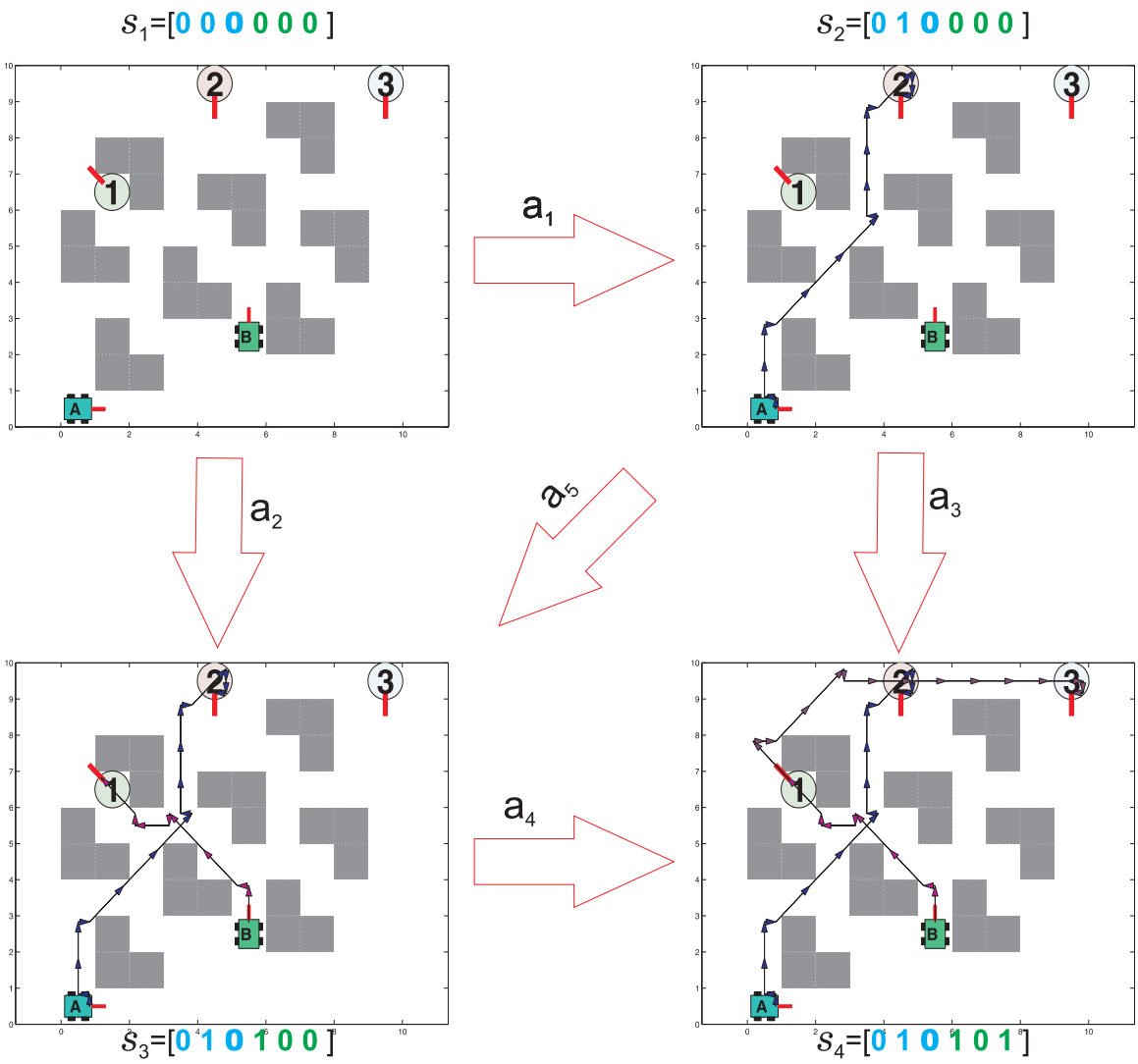


Figure 5.5: *The top left figure shows an example of a scenario in which two robots (A and B) should serve three goals(1, 2 and 3). The question is which robot should serve which goal and in what order. In the figure four possible plans* $s_1 - s_4$ *are drawn. In the plan* s_2 *(top right) only one goal is served while in plan* s_4 *all goals are served. Performing each plan incurs some costs and also receives some rewards. However, we are looking for the optimal solution.*

Algorithm POPTR(R,Θ)

A: Find The Optimal Paths

for all robots

for all goals

Calculate the Optimal path by solving equation (5.11)

end do

end do

B: Find The Optimal Plan

Generate the optimal plan by solving equation (5.22)

Evaluate the plans and find the optimal plan

Generate the optimal plan for each robot

Table 5.2: *The algorithm to plan an optimal path for a team of robot*

moving from plan s_i to s_j. Therefore R is defined as:

$$R : S \times A \times S \rightarrow \mathbb{R} \tag{5.24}$$

Based on the above formulation, there is a similarity between the MDP defined in Section 5.1.1 and our problem. The four tuple S, A, T, R we defined here is very similar to a MDP framework. There are similarities between S and S, A and A, T and T and R and R. Therefore, we could use the MDP solution to solve our problem.

5.4 The Algorithm

The algorithm to plan an optimal path for a team of robots (POPTR) is explained in Table 5.2. To run the algorithm we need to know the following information: R and Θ. R is a set which contains the pose of robots and objects. This information is provided by the networked cameras. We also need to know Θ which is set of all criteria. The set contains all of the parameters β's and γ's. The parameters are set by the CS based on the environment conditions and needs. For example, if the CS observes a fire and wants to check the situation closely, it should send a robot that can reach the position as soon as possible. Therefore, it sets β_L to zero but considers a high β_T. It also considers a high Γ_T. Having the information, the algorithm finds the optimal path for each robot to the goal. This is done by solving MDP. After finding the optimal individual paths, it is the time to find the optimal plan for the team. There is some information (e.g., traveling time, goal reward) that is generated in the previous stage and is needed for this stage. First, we generate the feasible plans and after that an MDP is run to find the optimal plan sequence.

We use value iteration (Section 5.1.1) to solve the MDP. One important issue with this method is the change in the environment. Since the environment is dynamic, we may experience changes in the environment, e.g., an unforeseen obstacle appears on the robot path and blocks it. In this case we recalculate the value function. As we have a good initial starting value for value functions, in a few iterations the algorithm will converge. Because of that we call this method *active* cost-reward based robot guidance since we can change the optimal path according to changes in the environment and also our needs. Besides that, assuming all team members (the robots) have access to Θ, using POPTR, they, individually, could find a unique solution to the problem. Therefore, either the CS could run

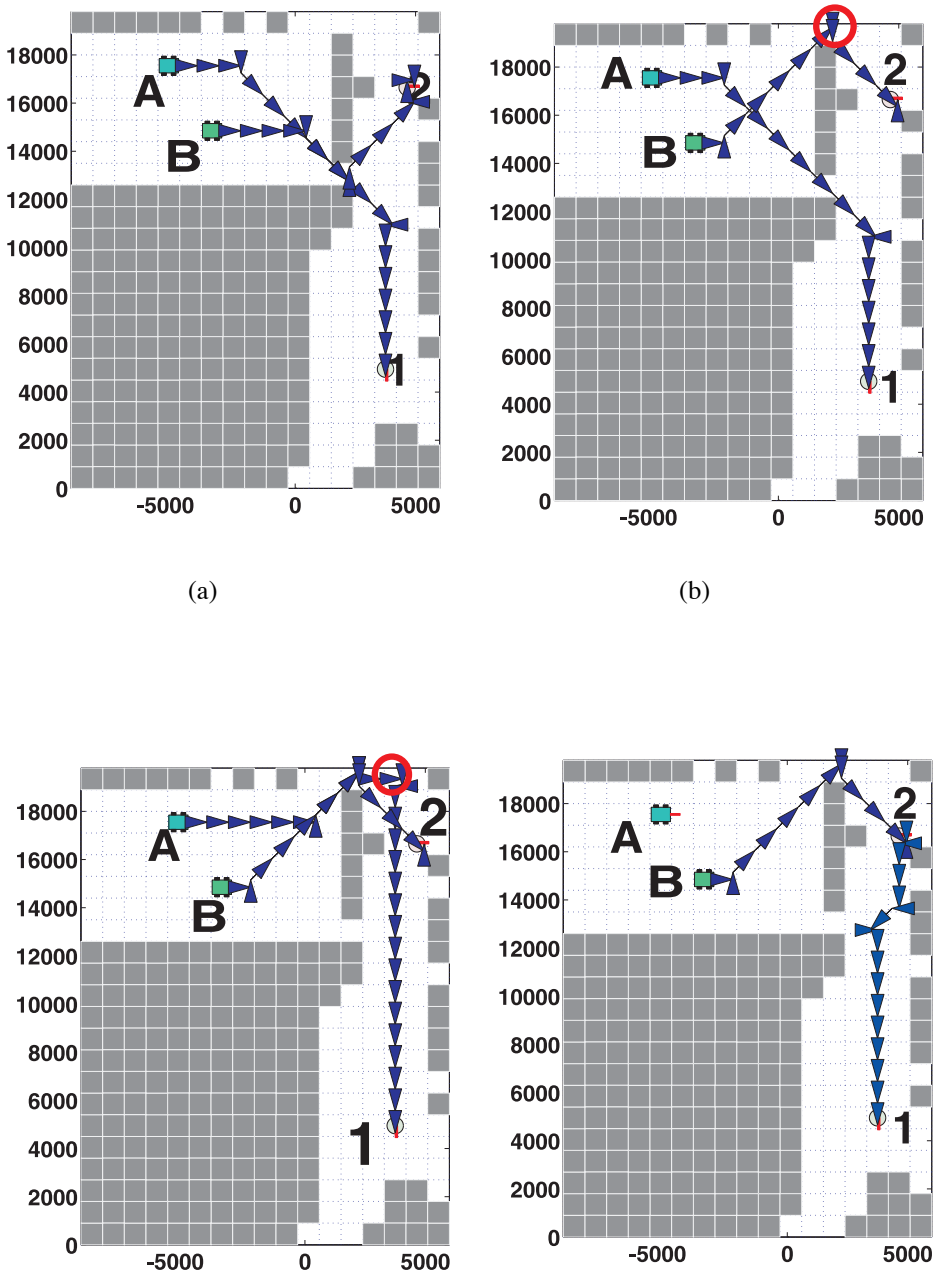
the algorithm and send the decision (centralized) or each robot separately (decentralized), could do it.

5.5 Experiments

Here, first we run a series of simulations in which we play with the parameters in such a way that for similar robots and goals positions different paths are generated. Afterwards, we run a series of real world experiments.

5.5.1 Simulations

In Fig. 5.6 a number of generated paths for different robots and goal positions are drawn. Our lab is considered as the simulation environment. The discretized space is shown in the figure. There are two robots (A and B) ready to serve the goals (1 and 2). Each cell size is $90cm \times 90cm$. Fig. 5.6(a) shows the first scenario when all coefficients are set to zero. This causes the robots to reach the goals as fast as possible. Fig. 5.6(b) shows a similar scenario. The only difference is that robot B should quicker reach a point where goal 2 is visible. For this goal, we set α_T to 1 and visibility coefficient to 5. Comparing to the first scenario, robot B sooner reaches a point (marked by a circle) where goal 2 is visible. However, it takes a longer path. Fig. 5.6(c) shows a different path from other two. This time, we set α_T for the first goal to 1 (still keep $\alpha_d = 0$) and visibility coefficient to 5. This time robot A reaches a point (marked by a circle) sooner where there is goal 1 is visible. Fig. 5.6(d) shows a scenario where a robot is asked to be kept in standby mode (for possible future use). Therefore a single robot should serve the two goals. Therefore, it keeps robot A in standby mode and deploy robot B to serve the goals.



(a) (b)

(c) (d)

Figure 5.6: *The figures represents the discretized environment of our lab. The dark states are obstacles and the white states are the free states. There are two robots (A and B) available to serve the two goals (1 and 2). There are 4 different scenarios are considered. Based on the the situation, different sets of parameters are used and as a result different paths are generated.*

5.5.2 Real NRS scenario

Also, we ran a real-robot experiment in our lab. There are 10 ceiling-mounted fixed cameras, each partially observes part of the field. Parts of the lab are covered by more than one camera. On the other hand, there are areas which are not covered by any of the cameras. The cameras are networked. Each camera in the network has a frame rate of 30 fps, and a resolution of 640×480 pixels. A background subtraction algorithm is used to detect the robot positions. A 2-D Gaussian observation model is considered for the cameras. More details on the experimental setup are provided in [9]. The robot is equipped with a Sick laser scanner. The data from the cameras are fused using Weighted Linear Combination of observation (WLC)[7]. A Kalman filter is used to track the robot using the cameras fused data and also odometry.

Figs. 5.7, 5.8 and 5.9 show the generated path, estimated path and real path for 3 different sets of parameters. The real path is the path estimated using the Sick laser scanner and used as the ground truth. The generated path is marked by the arrows. Each set of parameters is considered for a different situation. Fig. 5.7 shows a scenario in which the shortest path is desired. To do so, we set all parameters to zero. When a robot should reach the goal position in the shortest possible time and should not care about other parameters (e.g., an emergency mission), such a set of parameters should be used. If improvement in the robot localization uncertainty is desired the robot should take a path which has better camera coverage. This can be done, by assigning a higher priority to the localization certainty reward. Then, the robot takes a path which has the largest camera coverage but it is longer comparing to the previous path. Fig. 5.8 shows such a scenario. The rectangle drawn in the figure shows the area which is undercover of the cameras 3,4 and 10. In this

Figure 5.7: *The figure represents the first scenario. In the figure the planned, estimated and real paths of the robot are drawn. The figure shows the paths while reaching the goal in the shortest time is the concern.*

scenario, only, the value of localization certainty reward is set to 5 and the other parameters are similar to the first scenario. The robot is guided to the area which is covered by the cameras. However, as we can see from the figure, the robot takes a path which is very close to the obstacle. By giving a similar priority to the maneuvering reward (sets to 5), the robot takes a different path comparing to other two scenarios. The path is drawn in Fig. 5.9. It is longer than the path in the first scenario and has less camera coverage comparing to the second scenario. However, the robot takes the path which is farther from the obstacles around.

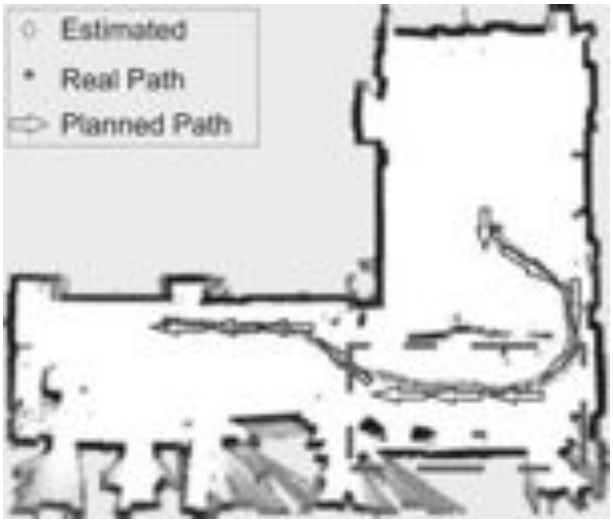


Figure 5.8: *The figure represents the second scenario. In the figure the planned, estimated and real paths of the robot are drawn. The figure shows the paths when the localization uncertainty is given a relative larger priority comparing to the other parameters.*

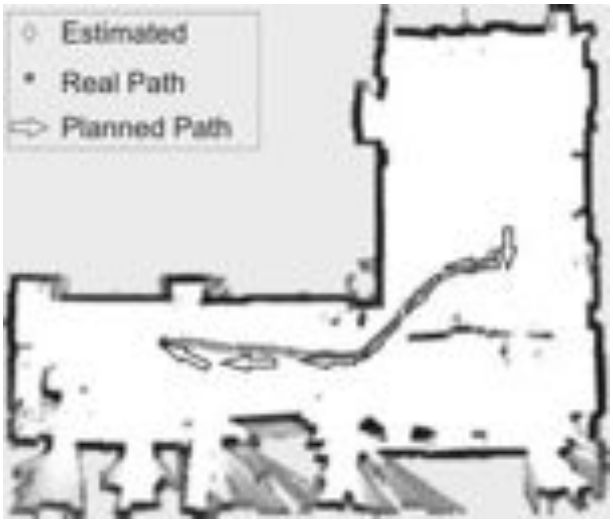


Figure 5.9: *The figure represents the third scenario. In the figure the planned, estimated and real paths of the robot are drawn. By assigning priority to the maneuvering as well as the localization, a different path is generated which is shown in the figure.*

Chapter 6

Conclusions and Future Work

6.1 Cooperative Perception

In this thesis, we introduced two novel algorithms: Cooperative Opinion Pool (COP) and p-norm Opinion Pool (POP) for fusion of probabilistic sensors. We investigated the performance of the COP algorithm in a multi-robot simulated environment. Simulations (Subsection 3.5.1.4) show that entropy of robot and localization decreases with the number of cooperating robots. We also compared the COP with Linear Opinion Pool (LOP) and Logarithmic Opinion Pool (LGP), two popular probabilistic sensor fusion methods for a simulated case study. The simulation is designed to reflect a real environment and results are satisfactory. COP could handle the disagreement among the observations. The conflict between the observation can be resolved by automatically recognizing and removing the faulty observations. The computational cost of COP is more than LOP and LGP. However, it can be reduced by considering only the first two component.

Following that, we introduced POP. We defined a series of specifications for evaluating a

fusion method. LOP and LGP, are evaluated based on those specifications. A new method to overcome the difficulties of LOP and LGP is introduced. This new method complies with most of the defined specifications. The computational cost of POP is compared 3.5.1.2 to the other two. Also an algorithm is provided to set the POP parameters. Based on the simulations and experimental results, we can conclude the POP is a reliable method and a good substitute for LOP and LGP.

As future work, we consider to use the method in a more complex environment. For the environment with large amount of states, we look for ways to approximate POP. One way is to distributedly compute POP. Another direction for future work is the intelligent selection of observations. Instead of considering the information of all sensors only part of them that have more accurate measurements can be chosen.

6.2 Active Cooperative Perception

The problem of Active Cooperative Perception was defined and formalized. We showed how to determine the best action and update the belief. Since updating the belief and choosing the optimal action in general can be computationally expensive and demand memory, an analytical solution for a linear system with Gaussian uncertainties is derived (Section 4.5). We also showed how we can consider a cost for implementing an action and integrate it in the formula. Following this, we ran a series of simulation to show the effect of ACP on reducing the uncertainty.

We have derived the formula for linear system with Gaussian noise. However there are other special cases (e.g., linear system with non-Gaussian noise) that can be considered. As many systems are nonlinear and/or the system noise distribution is not Gaussian, ap-

proximation of the action selection mechanism and belief update can provide a solution for such systems. One way can be to estimate the action selection formula using particle filter.

6.3 ACP and Cooperative Path Planning

The problem of generating an optimal path for several specifications for a team of robots was considered. We translated our objective into costs and rewards. A MDP framework is used to address the problem which allows to prioritize the different objectives in a flexible way by changing the reward coefficient. We also can solve the MDP in real time using value iteration. The problem is solved in two stages. First we determine the optimal paths based on our objectives. Afterward, the information is used to calculate the optimal plan.

Although in the URUS [73] project experiments around 10 robots were used and the algorithm is fast enough for this number of robots, as the number of robots and goals increases, the computational complexity can be a burden. Therefore, for the future work, we focus on ways to tackle the computational complexity of the algorithm. This can be done by approximating the MDP solution [27, 34, 65], the algorithm and/or distributedly solve the problem. Parallel processing can provide some helps as there are some processing sources available.

Bibliography

[1] A. Adjoudani and C. Beno. On the integration of auditory and visual parameters in a hmm-based asr. *NATO ASI: Speech reading by Humans and Machines*, pages 161–189, 1996.

[2] S. Alag, A. M. Agogino, and M. Morjaria. A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics. *AI EDAM*, 15(4):307–320, 2001.

[3] B. Argall, B. Browning, and M. Veloso. Learning robot motion control with demonstration and advice-operators. In *In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008)*, 2008.

[4] A. Arsénio and M. I. Ribeiro. Active range sensing for mobile robot localization. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, Washington, DC, USA, 1998. Proceedings of the 1998 IEEE/RSJ.

[5] A. Bakhtari and B. Benhabib. An active vision system for multitarget surveillance in dynamic environments. *Systems, Man, and Cybernetics, Part B, IEEE Transactions on Publication*, 37(1):190–198, 2007.

[6] G. Banos, l. Guibas, J. C. Latombe, S. M. LaValle, D. Lin, R. Motwani, and C. Tomasi. Motion planning with visibility constraints: Building autonomous observers, 1998.

[7] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic Press Professional, 1988.

[8] H. M. Barber, M. V. Vela, A. G. Skarmeta, and M. Z. Izquierdo. Fuzzy rules for sonar and infrared sensors fusion. In *3th International Fusion Conference (FUSION2000), París (France)*, 2000.

[9] M. Barbosa, A. Bernardino, D. Figueira, J. Gaspar, N. Gonçalves, P. Lima, P. Moreno, A. Pahliani, J. Santos-Victor, M. Spaan, and J. Sequeira. Is-robot-net: A testbed for sensor and robot network systems. In *In Proc. of IROS 2009*, 2009.

[10] M. Barbosa, A. Bernardino, D. Figueira, J. Gaspar, N. Goncalves, P. U. Lima, P. Moreno, A. Pahliani, J. Santos-Victor, M. T. J. Spaan, and J. Sequeira. Isrobotnet: A testbed for sensor and robot network systems. In *IROS*, pages 2827–2833, 2009.

[11] J. A. Benediktsson, J. R. Sveinsson, O. K. Ersoy, and P. H. Swain. Parallel consensual neural networks. *TNN*, 8(1):54–64, January 1997.

[12] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. AS, 2nd edition, 2000.

[13] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their population distributions. *Bulletin Calcutta Mathematical Society*, 35:99–109, 1943.

[14] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Appearance based active object recognition, 2000.

[15] D. Campolo, L. Schenato, P. Lijuan, D. Xinyan, and E. Guglielmelli. Multimodal sensor fusion for attitude estimation of micromechanical flying insects: A geometric approach. *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3859–3864, Sept. 2008.

[16] A. Cassandra, L. Kaelbling, and J. Kurien. Discrete bayesian uncertainty models for mobile-robot navigation, 1996.

[17] W. K. Choi, S. J. Kim, and H. T. Jeon. Multiple sensor fusion and motion control of snake robot based on soft-computing. *Knowledge-Based Intelligent Information and Engineering Systems*, 3682(6):291–297, 2005.

[18] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, June 2005.

[19] A. C. S. Chung and H. C. Shen. Entropy-based markov chains for multisensor fusion. *J. Intell. Robotics Syst.*, 29(2):161–189, 2000.

[20] D. J. Cook, P. J. Gmytrasiewicz, and L. B. Holder. Decision-theoretic cooperative sensor planning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1013–1023, 1996.

[21] A. J. Davison and N. Kita. 3d simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In *CVPR*, pages 384–391, 2001.

[22] F. Deinzer, J. Denzler, and H. Niemann. Viewpoint selection: Planning optimal sequences of views for object recognition. In *CAIP03*, pages 65–73, 2003.

[23] J. Denzler and C. M. Brown. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(2):145–157, 2002.

[24] J. Denzler, M. Zobel, and H. Niemann. On optimal camera parameter selection in kalman filter based object tracking. In *Proceedings of the 24th DAGM Symposium on Pattern Recognition*, pages 17–25, London, UK, 2002. Springer-Verlag.

[25] J. Denzler, M. Zobel, and H. Niemann. Information theoretic focal length selection for real-time active 3-d object tracking. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 400, Washington, DC, USA, 2003. IEEE Computer Society.

[26] M. Dietl, J. S. Gutmann, and B. Nebel. Cs freiburg: Global view by cooperative sensing. In *In RoboCup 2001 International Symposium*, pages 133–143. Springer, 2001.

[27] T. G. Dietterich. An overview of maxq hierarchical reinforcement learning. In *SARA '02: Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation*, pages 26–44, London, UK, 2000. Springer-Verlag.

[28] M. Dittmer, J. Kibbel, H. Salow, and V. Willhoeft. Team-lux darpa urban challenge 2007. Technical report, June 1st, 2007.

[29] H. F. Durrant-Whyte. Sensor models and multisensor integration. *IEEE Transcations on Robotics and Automation*, 7(6):97–113, 1988.

[30] H. F. Durrant-Whyte and T. C. Henderson. Multisensor data fusion. In *Springer Handbook of Robotics*, pages 585–610. 2008.

[31] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A*, 95(25):14863–14868, December 1998.

[32] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[33] B. Fassinut-Mombot and J. Choquel. A new probabilistic and entropy fusion approach for management of information sources. *Information Fusion*, pages 35–47, 2004.

[34] A. Ferrein, C. Fritz, and G. Lakemeyer. Extending dtgolog with options. In *IJCAI*, pages 1394–1395, 2003.

[35] D. Figueira, P. Moreno, A. Bernardino, and J. Gaspar. Detection and localization of persons and robots in a networked camera system. Technical report, 2009.

[36] D. Figueira, P. Moreno, A. Bernardino, and J. Gaspar. Detection and localization of persons and robots in a networked camera system. 2009.

[37] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 1998.

[38] D. Fox, W. Burgard, and S. Thrun. Markov localization for reliable robot navigation and people detection. In *Proc. of the Dagstuhl Seminar on Modelling and Planning for Sensor-Based Intelligent Robot Systems*, 1999.

[39] D. Fox, W. Burgardy, F. D. Sebastian, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots dieter fox, wolfram burgard. In *In Proc. of the National Conference on Artificial Intelligence (AAAI*, pages 343–349, 1999.

[40] E. Franklin and J. White. Data fusion lexicon. In *Technical Panel for C^3, Naval Ocean Systems Center*, pages 3721–3726, 1987.

[41] T. Fukuda, K. Shimojima, F. Arai, and H. Matsuura. Multi-sensor integration system based on fuzzy inference and neural network for industrial application. In *Proc. IEEE International Conference on Fuzzy Systems*, pages 907–914, 1992.

[42] R. Gayle, W. Moss, C. L. Ming, and D. Manocha. Multi-robot coordination using generalized social potential fields. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 3695–3702, Piscataway, NJ, USA, 2009. IEEE Press.

[43] B. Grocholsky and B. Grocholsky. Information-theoretic control of multiple sensor platforms, 2002.

[44] J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities*, pages 457–464. MIT Press, Cambridge, MA, USA, 1988.

[45] M. A. K. Jaradat and R. Langari. A hybrid intelligent system for fault detection and sensor fusion. *Appl. Soft Comput.*, 9(1):415–422, 2009.

[46] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *Proceedings of IEEE International Conference on Methods and Models In Automation and Roboticss, 2005*, pages 566–580. MorganKaufmann, 1997.

[47] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98, 1986.

[48] N. Kubota and K. Nishida. Cooperative perceptual systems for partner robots based on sensor network. *Int. J. of Computer Science and Network Security*, 6(11):19–28, 2006.

[49] S. Kullback. *Information Theory and Statistics*. Dover Books, New York, 2nd ed., 1968. first edition New York:Wiley, 1959, 1959.

[50] J. C. Latombe. *Robot motion planning*. Kluwer Academic Publishers, 1991.

[51] H. Liu, D. J. Brown, and H. Li. Parametric planning for multiple cooperative robots. *Journal of Intelligent and Robotic Systems*, 44(2):93–105, 2005.

[52] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White. Revisiting the jdl data fusion model ii. In *In P. Svensson and J. Schubert (Eds.), Proceedings of the Seventh International Conference on Information Fusion (FUSION 2004)*, pages 1218–1230, 2004.

[53] R. C. Luo, S. H. H. Phang, and K. L. Su. Multilevel multisensor based decision fusion for intelligent animal robot. In *ICRA*, pages 4226–4231, 2001.

[54] J. Manyika and H. Durrant-Whyte. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1995.

[55] J. Manyika and H. D. Whyte. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1995.

[56] X. Miao, M. R. Azimi-Sadjadi, B. Tan, A. C. Dubey, and N. H. Witherspoon. Detection of mines and minelike targets using principal component and neural-network methods. *IEEE Transactions on Neural Networks*, 9(3):454–463, 1998.

[57] S. Mitul. Multi-robot motion planning by incremental coordination. In *In IROS*, pages 5960–5963, 2006.

[58] R. R. Murphy. Dempster-shafer theory for sensor fusion in autonomous mobile robots. *IEEE Transcations on Robotics and Automation*, 14(2):197–206, 1998.

[59] A. Pahliani and P. Lima. Improving self localization and object localization by a team of robots using sensor fusion. In *Proc. of CONTROLO 2006*, page 336, 2006.

[60] A. Pahliani and P. U. Lima. Cooperative opinion pool: a new method for sensor fusion by a robot team. In *IROS*, pages 3721–3726, 2007.

[61] A. Pahliani, M. Spaan, and P. M. Lima. Decision-theoretic robot guidance for active cooperative perception. In *Conference on Intelligent Robots and Systems (IROS 2009)*, Octobr 2009.

[62] A. Pahliani, M. T. J. Spaan, and P. U. Lima. Fault-tolerant probabilistic sensor fusion for distributed multi-agent systems. In *Proc. of International Conference on Intelligent Robots and Systems*, 2010.

[63] L. E. Parker, B. Kannan, F. Tang, and M. Bailey. Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *In Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1016–1022, 2004.

[64] E. Parzen. On estimation of a probability density function and mode. *Computational Statistics and Data Analysis*, 1962.

[65] S. Proper and P. Tadepalli. Solving multiagent assignment markov decision processes. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 681–688, Richland, SC, 2009. International Foundation for Autonomous Agents and Multiagent Systems.

[66] H. Qu, S. Yang, X. Willms, R. Allan, and Z. Yi. Real-time robot path planning based on a modified pulse-coupled neural network model. *Trans. Neur. Netw.*, 20(11):1724–1739, 2009.

[67] V. Rigaud and L. Marce. Absolute location of underwater robotic vehicles by acoustic data fusion. In L. Marce, editor, *Proc. IEEE International Conference on Robotics and Automation*, pages 1310–1315 vol.2, 1990.

[68] A. Rogers, R. K. Dash, N. R. Jennings, S. Reece, and S. Roberts. Computational mechanism design for multi-sensor information fusion. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1463–1464, New York, NY, USA, 2006. ACM.

[69] M. Rosencrantz, G. J. Gordon, and S. Thrun. Decentralized sensor fusion with distributed particle filters. In *UAI*, pages 493–500, 2003.

[70] W. B. S. Thrun and D. Fox. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. MIT press, Upper Saddle River, NJ, USA, 2005.

[71] Y. M. Saiki, E. Takeuchi, and T. Tsubouchi. Vehicle localization in outdoor woodland environments with sensor fault detection. In *ICRA*, pages 449–454, 2008.

[72] G. Sanchez, C. Mxico, and J. C. Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *In IEEE International Conference on Robotics and Automation*, pages 2112–2119, 2002.

[73] A. Sanfeliu and J. Andrade-Cetto. Ubiquitous networking robotics in urban settings. In *Proceedings of the IEEE/RSJ IROS Workshop on Network Robot Systems*, 2006.

[74] B. Schiele and J. L. Crowley. Transinformation for active object recognition. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 249, Washington, DC, USA, 1998. IEEE Computer Society.

[75] G. Settembre, P. Scerri, A. Farinelli, K. P. Sycara, and D. Nardi. A decentralized approach to cooperative situation assessment in multi-robot systems. In *AAMAS (1)*, pages 31–38, 2008.

[76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976.

[77] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

[78] C. Smaili, M. E. B. E. Najjar, and F. Charpillet. Multi-sensor fusion method using dynamic bayesian network for precise vehicle localization and road matching. In *ICTAI (1) 2007*, pages 146–151, 2007.

[79] A. S. Solberg, G. Storvik, and R. Fjortoft. A comparison of criteria for decision fusion and parameter estimation in statistical multisensor image classification. In G. Storvik, editor, *Proc. IEEE International Geoscience and Remote Sensing Symposium IGARSS '02*, volume 1, pages 72–74 vol.1, 2002.

[80] F. Soong and A. Rosenberg. On the use of instantaneous and transitional spectral information in speaker recognition. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, volume 36, pages 871–879, 1988.

[81] M. T. J. Spaan. Cooperative active perception using pomdps. In *AAAI 2008 Workshop on Advancements in POMDP Solvers*, jul 2008.

[82] A. Stroupe, M. C. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *IEEE International Conference on Robotics and Automation, May, 2001*. IEEE, May 2001.

[83] A. W. Stroupe, C. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. pages 1092–1098, 2001.

[84] C. Sun, G. S. Arr, R. P. Ramachandran, and S. G. Ritchie. Vehicle reidentification using multidetector fusion. *IEEE Transactions on Intelligent Transportation Systems*, 5(3):155–164, 2004.

[85] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT, 1998.

[86] K. Sycara, a. B. Y. R. Glinton, J. A. Giampapa, S. R. Owens, M. Lewism, and L. C. Grindle. *An Integrated Framework for High Level Information Fusion*, April 2007.

[87] C. Tarin, H. Brugger, R. Moscardo, B. Tibken, and E. Hofer. Low level sensor fusion for autonomous mobile robot navigation. In *Instrumentation and Measurement Technology Conference, 1999. IMTC99*, pages 1377–1382, March 1999.

[88] A. Telmoudi and S. Chakhar. Data fusion application from evidential databases as a support for decision making. *Information & Software Technology*, 46(8):547–555, 2004.

[89] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artif. Intell.*, 99(1):21–71, 1998.

[90] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, September 2005.

[91] K. Tischler, C. Duchow, and B. Hummel. Information fusion for cooperative vehicles. In *GI Jahrestagung (1)*, pages 374–378, 2006.

[92] P. Tournassoud. A strategy for obstacle avoidance and its application to multi-robot systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Octobr 1986.

[93] N. Ukita and T. Matsuyama. Real-time multi-target tracking by cooperative distributed active vision agents. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 829–838. ACM, 2002.

[94] N. Ukita and T. Matsuyama. Real-time cooperative multi-target tracking by communicating active vision agents. *Comput. Vis. Image Underst.*, 97(2):137–179, 2005.

[95] S. Vasudevan, F. T. Ramos, E. Nettleton, and H. F. Durrant-Whyte. Heteroscedastic gaussian processes for data fusion in large scale terrain modeling. In *ICRA*, pages 3452–3459, 2010.

[96] K. F. Wallis. Combining density and interval forecasts: A modest proposal. *Oxford Bulletin of Economics and Statistics*, pages 983–994, 2005.

[97] H. Wang, G. P. K. Yao, and D. Estrin. Entropy-based sensor selection heuristic for target localization. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 36–45, New York, NY, USA, 2004. ACM.

[98] K. H. C. Wang and A. Botea. Tractable multi-agent path planning on grid maps. In *IJCAI*, pages 1870–1875, 2009.

[99] A. L. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[100] A. L. Zadeh. Fuzzy algorithms. *Information and Control*, 12(2):94–102, 1968.

[101] Y. Zhou and H. Leung. Minimum entropy approach for multisensor data fusion. In *SPWHOS '97: Proceedings of the 1997 IEEE Signal Processing Workshop on Higher-Order Statistics (SPW-HOS '97)*, page 336. IEEE Computer Society, 1997.