UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

An Integrated Bayesian Approach to Multi-Robot Cooperative Perception

Aamir Ahmad

Supervisor: Doctor Pedro Manuel Urbano de Almeida Lima

Thesis specifically prepared to obtain the PhD Degree in Electrical and Computer Engineering

Draft

March 2013

This thesis is dedicated to my late grandmother Mrs. Tabassum Yunus who will be the source of eternal inspiration for me.

Abstract

This thesis introduces novel methods for cooperative perception (CP) in a multirobot team. CP involves cooperatively estimating the position of a tracked object by a team of mobile robots or/and cooperatively estimating the localization of the tracking robots in the team. The central feature of these methods is their integrated design, where different issues, such as sensor fusion, disagreement among sensors, inconsistent sensor measurements, occlusions and sensor failures, are handled within a single Bayesian framework. The introduced CP methods are based on particle filters and graph-based optimization, and address scenarios where: i) either the individual robot's localization or the object tracking by each individual robot is assumed to be determined by other existing methods; and ii) the tracking robot's pose, teammates' poses and the tracked object's positions are estimated in a unified framework.

The theoretical formulation of the CP methods and the results of their implementation on real robots are presented. The case studies compare the features and provide quantitative analysis of the introduced methods, including robustness tests, when applied in real robot experiments. We also demonstrate the utilization of one of the CP methods as a feedback module in a closed loop multirobot formation control system that minimizes the uncertainty of the cooperatively tracked object's position estimate.

The prerequisites of the experimental evaluations led to the development of a novel detection algorithm for spherical objects, the implementation of a stereo vision-based ground truth system and the creation of real robot datasets on which the proposed CP methods were implemented. The datasets are made publicly available for the benefit of the robotics community at large.

Keywords: Multirobot Systems, Visual Tracking, Cooperative Object Tracking, Cooperative Robot Localization, Particle Filters, Sensor Fusion, Graph Optimization, Moving Landmarks, Spherical Object Detection, Extended Kalman Filter.

Resumo

Nesta tese são apresentados novos métodos de percepção cooperativa (PC) numa equipa com múltiplos robôs. A percepção cooperativa envolve estimar (cooperativamente) a posição de um objecto monitorizado por uma equipa de robôs móveis e/ou estimar a localização dos robôs que executam a monitorização. A característica central destes métodos é a abordagem integrada, onde diferentes problemas, tais como a fusão de dados sensoriais, o desacordo entre sensores, os dados sensoriais inconsistentes, as oclusões e as falhas de sensores, são tratados através de um enquadramento Bayesiano único. Os métodos de PC propostos são baseados em filtros de partículas e optimização baseada em grafos, e são concebidos para cenários em que: i) ou a localização de cada robot ou a monitorização de objectos por cada robot é assumida como sendo determinada por outros métodos existentes, e ii) a postura (posição + orientação) do robot que executa a monotorização, as posturas dos seus companheiros de equipa e a posição do objecto monitorizado são estimadas usando um enquadramento unificado.

A formulação teórica dos métodos de PC e os resultados da sua aplicação em robôs reais são também introduzidos. Os estudos de caso visam comparar as características dos métodos introduzidos e fornecer uma análise quantitativa dos mesmos, incluindo testes de robustez, quando aplicados em experiências com robôs reais. Também se demonstra a utilização de um dos métodos de PC como um módulo de retroacção num sistema de controlo de formações de múltiplos robôs, em cadeia fechada, que minimiza a incerteza da posição estimada do objecto seguido cooperativamente.

Os pré-requisitos das avaliações experimentais levaram ao desenvolvimento de um novo algoritmo para detecção de objectos esféricos, à implementação de um sistema de *ground truth* baseado em visão estéreo, e à criação de conjuntos de dados de robôs reais em que os métodos de PC propostos foram implementados. Estes conjuntos de dados estão disponíveis para o benefício da comunidade de robótica em geral.

Palavras-chave: Sistemas de Multiplos Robôs, Seguimento Visual, Seguimento Cooperativo de Objecto, Localização Cooperativa de Robôs, Filtros de Partículas, Fusão de Sensores, Optimização de Grafos, Marcos Movéis, Detecção de Objectos Esféricos, Filtros de Kalman Extendidos.

Acknowledgements

This doctoral thesis would not have been possible without the support of all the loving and caring people around me.

First and foremost, I would like to express my immense gratitude to my supervisor Prof. Dr. Pedro U. Lima. Without his invaluable guidance and direction, it would not have been possible for me to reach this stage of my academic life. His insights, knowledge and scientific approaches have carried me not only through the course of my graduate studies but also will guide my future. To him and to his esteemed supervision, I will forever remain indebted.

I would like to thank Prof. Dr. José Santos-Victor, Prof. Dr. João Xavier and Prof. Dr. Rodrigo Ventura for their encouragement, support and valuable suggestions. The long brainstorming sessions that I could have with them was incredibly helpful for the completion of this thesis. I would also like to extend my gratitude to all the staff at Institute for Systems and Robotics Lisbon who have directly or indirectly been a great support during my stay there. To their ever assuring support, I will always remain indebted.

From the robotics labs of *Instituto Superior de Engenharia do Porto* and *Faculdade de Engenharia da Universidade do Porto*, I would like to thank Prof. Dr. Eduardo Silva, Prof. Dr. António Paulo Moreira and Prof. Dr. Luís Almeida for their encouragement and advices on varied technical subjects. To their ever welcoming gestures during my numerous visits to their labs, I will always remain indebted.

I would like to thank Prof. Dr. Wolfram Burgard for giving me an amazing opportunity to do research under his honored guidance at the research lab for Autonomous Intelligent Systems, *Albert-Ludwigs-Universität*, Freiburg. I am extremely obliged to Dr. Gian Diego Tipaldi for his inestimable amount of guidance, support and advices during my stay in Freiburg. To their hospitality and in-depth technical advices, I will always remain indebted.

My friends and colleagues at Institute for Systems and Robotics Lisbon, *Instituto Superior de Engenharia do Porto* and *Faculdade de Engenharia da Universidade do Porto*, André Dias, Bruno Lacerda, Bruno Nery, Carlos Neves, Danesh Tarapore, David Afonso,

Filipe Valente, Gonçalo Neto, Henrique Silva, Hugo Costelha, João Messias, João Reis, José Nuno, José Carlos, Luís Oliveira, Meysam Basiri, Stefan Witwicki, Tiago Nascimento, Tiago Veiga and many more have given me immense support which was indispensable for the completion of this thesis. To their friendliness and selfless support, I will always remain indebted.

No words can ever fathom the love and care my mother and father have always put into me. Thousands of miles away from them, I am blessed everyday by their utmost warmth and support. Their sincerest efforts to bring me up, educate me and instill in me the quest for knowledge is unparalleled. To their silent struggle throughout their lives to give the best to me, I will always remain indebted. The smile and the cheering words of my younger sister has seen me through some of the hardest times. To her kindness and selfless love, I will always remain indebted.

I am immensely grateful to my fiancee, Daniela, who has been a source of unlimited inspiration and motivation throughout the course of my graduate studies. Her care and support has seen me through some of my most dispirited times. To her extreme patience and unbounded love, I will always remain indebted.

My grandparents have been incomparable icons of inspiration, encouragement and wisdom for me. Their selfless contribution in bringing me up during a significant part of my childhood and inspiring me to reach limitless heights in quest for knowledge and education cannot be expressed in words. It is indeed very unfortunate that they could not live to see the completion of my doctoral thesis. To their immeasurable dedication, I will forever remain indebted.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The field of sensor fusion, including its use for single and multiple target tracking [1] [2] [3] [4] [5] [6] [7] is now very mature. However, it does frequently address situations where the sensors are static, know their location in a global frame with no uncertainty, and occlusions occur rarely. When sensors are mobile, e.g., mounted on the top of mobile robots, their knowledge of their own localization may degrade over time and/or during time periods due to a number of reasons (e.g., absence of known environment features, bad odometry) and this impacts the uncertainty in the determination of the target position in the global frame, where it is fused with the estimates from the other sensors. Furthermore, occlusions can occur more frequently, as they are due not only to the target object(s) path but also to the motion of the different sensors/robots. Therefore, the problem of cooperatively detecting and tracking a moving object by a team of mobile sensors is an extension of sensor fusion in which one has to handle occlusions, disagreements between sensors, and dynamic changes of the observation models due to frequent spatial changes. The inconsistency or disagreement among various mobile sensor observations can occur primarily due to the following factors: i) difference in each sensor's observation model ii) global localization uncertainty of the robots, the mobile platforms on which the sensors reside. Developing a Bayesian approach to cooperatively track an object by a team of mobile robots and use it to improve the tracking robots' localization, where the aforementioned inconsistencies or disagreements are addressed in an integrated manner is the central theme of this thesis.

1.2 Contributions

The novel contributions of this thesis are as follows:

- A particle filter-based (PF) algorithm for unified cooperative multi-robot localization and object tracking (PF-UCLT) in an integrated framework. The framework is decentralized and designed for online implementation on real robots.

- A pose graph optimization-based method for cooperative multi-robot localization and object tracking in an integrated framework. The framework is centralized and designed for offline implementation on previously collected robots' measurements datasets. It has been accepted as a full length-article [8] to be published in the proceedings of the *2013 IEEE International Conference on Robotics and Automation (ICRA 2013)*.

- A PF-based multirobot cooperative object tracking (PF-MCOT) algorithm which fuses the information from teammates by taking into account their localization confidences to reduce the tracked object's position estimate uncertainty while effectively handling inconsistent global frame observations of the poorly localized robots in the team. The algorithm is decentralized and designed for online implementation on real robots. It was initially published in the *European Conference on Mobile Robotics (ECMR 2011)* [9]. It was later voted as one of the best papers presented in the conference and invited for a special issue article in the *Robotics and Autonomous Systems (RAS)* journal, where it was accepted for publication [10]. PF-MCOT was successfully applied as a feedback module in a closed loop multirobot formation control system that minimizes the uncertainty of the cooperatively tracked object. This has been accepted as a full length-article [11] to be published in the proceedings of the *2013 IEEE International Conference on Robotics and Automation (ICRA 2013)*.

- A PF-based multirobot cooperative robot localization (PF-MCRL) algorithm where a visually shared object is used for regaining the poorly localized robot's correct localization. The algorithm is decentralized and designed for online implementation on real robots. It was published as a chapter in the book *RoboCup 2010: Robot Soccer World Cup XIV. Lecture Notes In Artificial Intelligence, Lecture Notes in Computer Science* [12].

- A mathematical proof for a bijection principle stating that the 3D position of a spherical object of known radius (the object tracked in the real robots experimental

evaluation of the CP approaches mentioned above) can be uniquely detected using only a single image from a fish-eye lens-based camera. It has been submitted as a full length article to the *Journal of Real-Time Image Processing: Special Issue on Robot Vision* and is currently under review.

1.3 State of The Art

Object tracking is a field of research where multiple techniques are currently being researched and developed extensively [13]. PFs are one of the most popular methods employed for tracking [14]. PF is a non-parametric filter. Non-parametric filters can efficiently handle multi-modal beliefs. In a generic tracker the motion model of the object being tracked can be completely unknown and might change over time hence using a parametric filter can lead to failures quite often. If one uses any standard motion model for the object in a parametric filter, the tracker can quickly result in low confidence on the posterior when the object motion changes to a different one or switches randomly. This makes it essential to have beliefs with multiple modes scattered over the whole state space which makes the use of a non-parametric filter appropriate. An interesting approach of fusing the Extended Kalman Filter (EKF) and Monte Carlo PF has been described in [15] where an integrated self-localization and ball tracking method is presented. In [16] a method for simultaneously estimating ball position and velocity using Monte Carlo Localization (MCL) is developed. An efficient implementation of Rao-Blackwellised PF which was successfully demonstrated on Sony AIBO robots in the four-legged league of RoboCup is presented in [14]. None of these works use the information from more than one sensor/robot, therefore being less robust to occlusions and very dependent on the relative state of the robot and the object tracked.

The authors in [6] presented an efficient solution for multiple static platforms tracking a moving target while the authors in [17] introduced an interesting approach for the case of a single moving platform tracking multiple moving moving targets. One important focus of this thesis is to combine significant parts of both these challenges under a common unified framework, i.e., to track a moving target using multiple moving sensor platforms. Including multiple targets with known data association can be viewed as its straightforward extension. However, the case of multiple moving platforms tracking multiple targets that occur with unknown data association is not addressed in this thesis but identified as one of the most prominent future work.

In [18], relationships between fixed world objects and moving objects is explored for

global object localization. These relationships are communicated to teammates where they form a set of constrained relations, solving which gives object location estimates. The authors in [7] present a cooperative PF based tracker for Sony AIBO robots, where the fusion of information involves communicating a reduced set of particles between the robots over the wireless network, which still remains a huge data set causing inefficient communication. Our approach overcomes this problem, explained in the subsequent chapters of this thesis. In [19] a new cooperative perception architecture is developed and tested on multiple UAVs for forest fire detection and localization. A substantial effort is put on developing the fire detector and fusion of data from various sensors used on-board a single and multiple UAVs. The errors that creep in due to the self-localization of the UAVs themselves are unaccounted for. They are addressed in this thesis.

In [4],[5] a decentralized PF for multiple target tracking is developed and deployed on flight vehicles. The communication bandwidth problem is solved by transforming the particle set into a Gaussian mixture model (GMM). In our approaches, we communicate only the observation measurement vectors and an associated noise covariance matrix between two robots. This not only further reduces the bandwidth usage but also prevents the recursive propagation of the estimation errors to the teammates. This problem occurs when sharing particles (or a parametrized form of it) among the teammates. Recently in [1], Tim Bailey et al. presented an efficient mechanism for cooperative robot localization which is an alternative method to overcome the recursive propagation of errors. This is done by centralizing the cooperative estimation. In their method, each robot first estimates its own pose and then communicates it, along with an inter-robot measurement, to a central data server. These data, received from all the robots in the team, are then fused to compute consistent relative localization of every robot. However, this implicitly means that the robots need to measure the relative distances to other robots in the team. This assumption could often be implausible, e.g, a scenario where the distance between the teammates is too large or the environment is too cluttered to use line-of-sight-based methods for inter-robot measurements.

In [20], a PF based tracker is presented with a unique and novel 3-D observation model based on color histogram matching. Each robot has an individual tracker and its most notable feature is that the tracking could be performed in 3-D space without the object color information, but at the cost of a heavy computational expense. In order to solve this issue, a novel 3D detector for spherical objects is developed. In [21], a sensor fusion technique for cooperative object localization using particle filters is presented. Parameters of a GMM approximating a teammate's tracker's particles are communicated to the other robots. Particles at a robot's tracker are then sampled using own belief and the received

GMM.

Another important focus of this thesis is on cooperative localization using a commonly tracked object by a team of robots. Self-localization is one of the most relevant topics of current research in robotics. Estimation-theoretic approaches to self-localization, as well as to self-localization and mapping (SLAM) have produced significant results in recent years, mainly for single robots, providing effective practical results for different applications. One of the research frontiers in this topic concerns now cooperative localization (and possibly mapping) using a team of multiple robots [22] [23] [24] [25] [26].

One of the earlier works on cooperative localization [27] addresses cooperative localization within a Kalman filter framework, where the relative positions of the robots are the observations of the filtering part of the algorithm, and the state includes the positions of all the robots. Fox et al introduced an extended version of the general Markov Localization algorithm [28], where two robots use measurements of their relative distance and bearing to insert an extra step in the belief update algorithm based on the Bayes filter. They used the Monte Carlo Localization (MCL) sampled version of Markov Localization algorithm to influence the weights of the particles of the observed robot from the particles sampling the inter-robot distance and bearing measurement model of the observing robot. Other authors address multi-robot localization using similar approaches, so as to provide relative localization of the team members in one of the team robots local frame from inter-robot distance measurement [29],[30]. None of these works uses environment information commonly observed by the team robots to improve their localization.

Other works attempt to take advantage of environment features and landmarks to help a multi-robot team to improve the pose estimates of its own team members, while simultaneously mapping the landmark locations. Fenwick et al [31] focus on convergence properties and performance gain resulting from the collaboration of the team members on concurrent localization and mapping operations. Jennings et al [32] describe a stereo-vision-based method that uses landmarks whose location is determined by one of the robots to help the other robot determining its location. The first approach addresses a general model that does not take advantage of particular features of the estimation-theoretic methods used (e.g., particle filters) to improve the robustness and to speed up cooperative localization, while the second is focused on a particular application.

Some of the subsequent chapters in this thesis include a brief section describing the state of the art relevant to the algorithms described therein.

1.4 Thesis Overview and Organization

In this section, we present the structural organization of this thesis. Figure 1.1 presents a graphical overview for the same[1].

In the cooperative perception algorithms presented here, object tracking is performed in 3D. However, it is not obvious that object detection in 3D is possible using only a dioptric vision system consisting of a single fish-eye lens-based camera. Therefore a parallel work on 3D object detection using a single dioptric camera system was done. The 3D detector and the principle on which it is based, along with its mathematical proof, is presented in Chapter 2.

In Chapter 3, a cooperative approach for tracking a moving object, by a team of mobile robots equipped with sensors, in a highly dynamic environment, is introduced. The tracker's core is a PF, modified to handle, within a single unified framework, the problem of complete or partial occlusion for some of the involved mobile sensors, as well as inconsistent estimates in the global frame among sensors, due to observation errors and/or self-localization uncertainty. Designed in a decentralized fashion, it fuses the object observation measurements from all the teammates at each robot while taking into account the teammate's localization uncertainties as a basis of assigning trust measure on their object observation measurements. This makes the cooperative tracking resistant to poor localization of teammates, occlusions and perception errors.

Chapter 4 describes a cooperative localization algorithm based on a modification of the Monte Carlo Localization algorithm. When a robot detects that it is lost, particles are spread in the state space not uniformly, but according to the information on the global location of a visually shared object obtained from a teammate, assuming that the same object is also visible to the lost robot. The lost robot receives the tracked global position of the visually shared object from the well-localized robots which have a high confidence on their estimate of the object's position. If the lost robot is also tracking the same object in its local frame, it uses this additional information to help re-localize itself. This algorithm assumes that the visually shared object is being tracked by each robot individually, using an object tracking method separate from the robot localization method.

In the first part of Chapter 5, a unified method is presented for cooperative target

[1]Please note that in Figure 1.1 and the rest of this thesis, detection refers to the process of classifying an object in a single frame without using any previous (in time) information about that object, whereas tracking refers to the process of estimating the position or/and velocity of the object based on the most recent detected position, its previously estimated position(s), object's motion model and the associated noise with them.

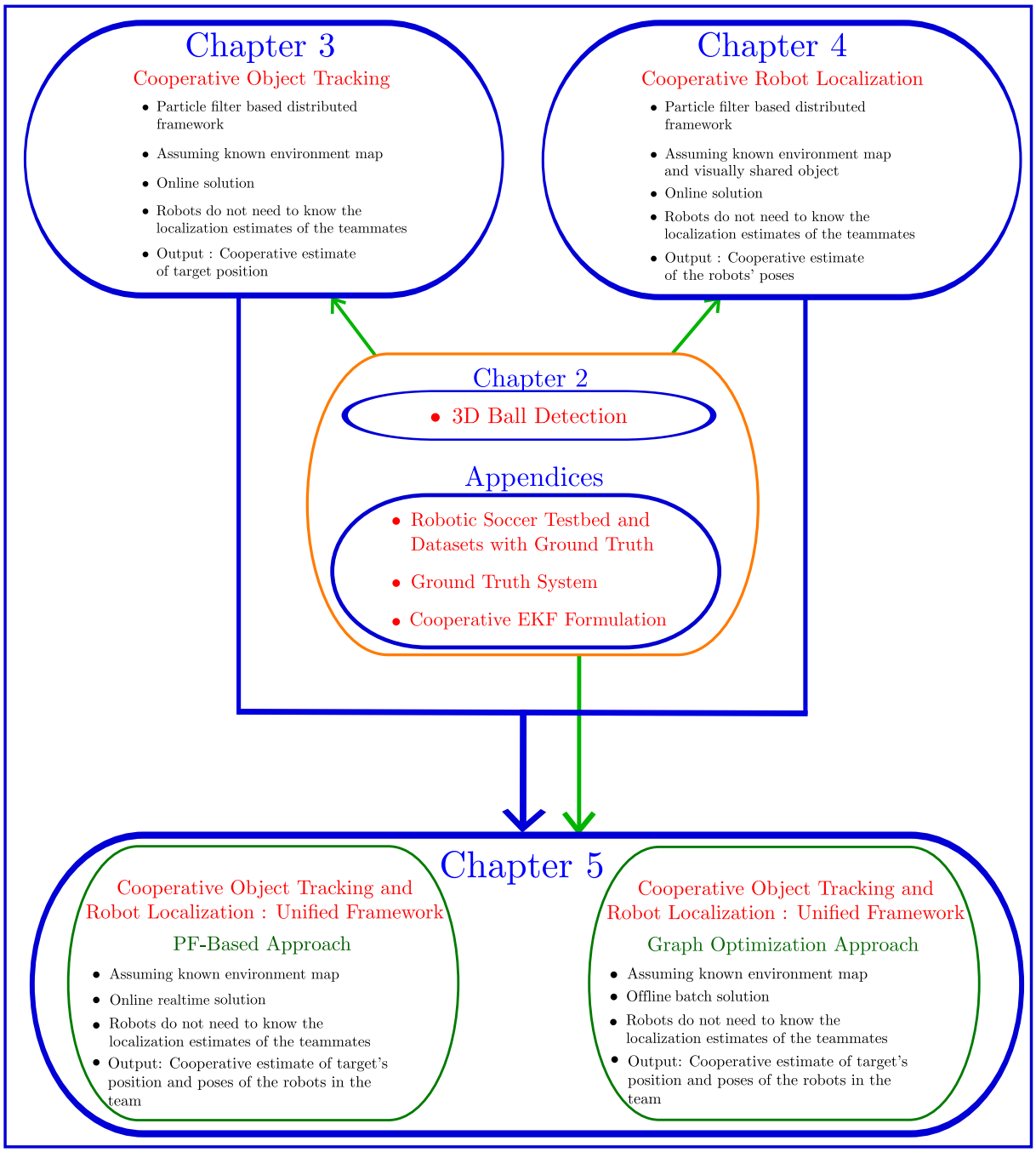


Figure 1.1: A flow diagram representing the organization of chapters in this thesis. Each blue box represents a separate chapter or appendix.

tracking from multiple moving robot frames while simultaneously localizing these robots using the tracked target as a moving landmark in addition to the known and static landmarks in the environment. We model this as a least squares minimization problem and

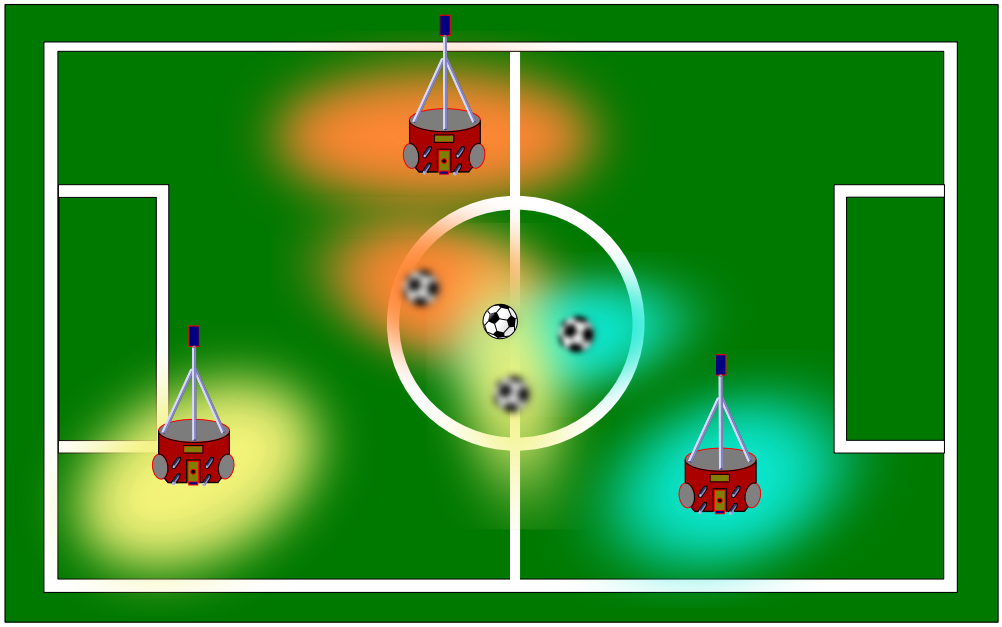


Figure 1.2: An Illustration of the testbed used in the case study of this thesis. Diffused colors around the robots represent the uncertainty in their position estimates. In addition to the ball, shown in its correct position, a blurred ball is shown at its measured position from each individual robot. Noise in that measurement is shown by diffused colors around that measured ball position.

show that it can be efficiently solved using sparse optimization methods. This is done by first constructing a pose graph representation of the problem, where the nodes are robots and target poses at individual time-steps, and the edges are their relative measurements. Static landmarks at known positions are used to define a common reference frame for the robots and the target. A least squares error function is derived from this graph and subsequently minimized by invoking an optimization solver. The g^{2}o [33] framework, which is used here for the graph optimization, is extended for the multi-robot localization and target tracking scenario. Note that, as opposed to the other CP frameworks developed in this thesis, this algorithm is an offline solver. It operates only on a previously collected dataset of observations and odometry and process all of it in a single batch.

The second part of Chapter 5 presents an online PF-based algorithm of a unified framework for multi-robot cooperative localization and object tracking. The algorithm works in a decentralized fashion, meaning each robot estimates its own pose, teammates's poses and the tracked object's position, which also corresponds to the state of each particle of the PF. At every time-step each robot operates on the odometry, static landmarks' obser-

vations and the target observation obtained by its own sensors as well as the ones obtained by its teammate's sensors and communicated to it using explicit wireless communication. Essentially, it unifies the algorithms developed in Chapter 3 and 4 while eliminating the problem of using any relative measurement twice, which would occur if the algorithms in Chapter 3 and Chapter 4 would run in parallel on a robot team. A novel method is proposed to substantially reduce the number of PF particles required for this algorithm, which otherwise would grow exponentially with the number of robots in a team.

Chapter 6 presents the results and analysis of a case study in which the unified CP algorithms developed in Chapter 5 are implemented on the same dataset for a fair comparison. To achieve this, a dataset was collected on the testbed, both of which are described in Appendix A. The analysis shows the algorithms' robustness to various scenarios, e.g, hardware problems and communication failures. The dataset is made publicly available[2] for the benefit of the robotics community at large. Various algorithms ranging from robot localization, object tracking to simultaneous localization and mapping (SLAM) can be implemented on this dataset.

The algorithm for multi-robot cooperative object tracking developed in Chapter 3 was implemented as the feedback module in a multi-robot closed loop formation control framework. This is presented as a case study in Chapter 7. Its goal was to accomplish the final objectives of the Portuguese project 'PCMMC: Perception Driven Coordinated Multi-Robot Motion Control[3]'. The project aimed to "conceive and implement an active approach to cooperative perception through coordinated vehicle motion control where the vehicle formation geometry would change dynamically so as to maximize the accuracy of cooperative perception of a static or dynamic target by the formation vehicles"[4]. One of its tasks dealt with the probabilistic methods for formation state estimation, co-operative localization and tracking of targets, while another task was focused on the integration of the cooperative tracking with the formation control algorithms on the soccer robots test-bed (Appendix A).

Since the implementation of the algorithms developed in this thesis are done on real robots, it becomes essential to fully describe the test-bed and the robot's architecture. This is presented in Appendix A. The ground truth (GT) system implemented for performance evaluation of the CP algorithms is presented in Appendix B.

[2]LRM Dataset download link: http://datasets.isr.ist.utl.pt/lrmdataset/4_Robots_DataSet/

[3](FCT PTDC/EEA-CRO/100692/2008)

[4](http://mediawiki.isr.ist.utl.pt/wiki/PCMMC:_Perception-Driven_Coordinated_Multi-Robot_Motion_Control)

Chapter 2

3D Spherical Object Detector

2.1 Introduction and Related Work

In recent years, omni-directional vision involving cata-dioptric [34] and dioptric [35] vision systems has become one of the most sought-after technology being employed in mobile robots. The primary reason of this success is a much wider field of view compared to a perspective projection lens-based camera. A dioptric vision system uses only a fish-eye lens camera instead of a perspective projection camera/parabolic mirror arrangement which is used in a cata-dioptric vision system. The calibration parameters of a cata-dioptric system are susceptible to variations, owing to physical vibrations and impact force when such a system is installed on a mobile robot platform and subjected to fast motion and unexpected collisions. The dioptric system overcomes this problem by having a more robust physical setup. Although we focus on a dioptric vision system (DVS) in this chapter, the algorithm developed here can be easily modified for cata-dioptric vision systems (CVS) by using the appropriate projection model and following the steps similar to the ones described further in this chapter.

Spherical object position estimation is a functionality required by innumerable applications in the areas ranging from mobile robotics, biomedical imaging, machine vision to material sciences. In mobile robotics, detection and tracking of spherical objects has gained substantial attention [36][37], including the use of omni-directional vision systems to do so. In [34], the authors present a circular Hough transform (HT) based spherical object detection using a CVS. Although they use a single camera to perform the detection in real-time, their algorithm assumes that the object maneuvers on a fixed and known ground plane hence the detection is performed in a 2-dimensional space. They extend the detection to the 3D space by using a stereo vision system (SVS) on their robots consisting

of a CVS and a perspective camera installed on the same robot [38]. A similar approach is taken by authors in [37], where an SVS comprising of a CVS and a perspective camera looking in the forward direction is used. Here, instead of using an HT-based detection, the authors use a set of heuristics such as the color of the projected blobs in the image, the roundness of the blobs, etc., to detect pixel position in each individual image from both the cameras and later process them using the SVS calibration parameters to obtain the spherical object's 3D position. The authors in [39] present an interesting method of using the rotational invariance of the spherical object in a structure tensor technique to remove the use of color in the detection process. However, they still use a single perspective camera and perform detection only in the 2D space, assuming that the spherical object is on a fixed and known ground plane. Furthermore, in all these works, the diameter of the spherical object is known beforehand, which is an acceptable assumption for a variety of mobile robot tasks.

Most of the existing methods in literature make use of an SVS to detect the spherical object's position in 3D even when its diameter is known beforehand. Our work however shows that if the diameter is known, then it is possible to perform the detection in 3D using only a single camera image. Some of the authors in our research group have earlier presented a method to do so using a color histogram mismatch-based algorithm [20], however the concept of uniquely detecting the 3D position using only a single camera image was not theoretically proven and the method was computationally heavy, often making it unsuitable for real-time applications.

In this work we present a spherical object detection algorithm for dioptric vision systems equipped with a fish-eye lens-based camera. The aim is to estimate the 3D world position of a spherical object's center using only a single image, given that the diameter of the object is known beforehand. Since it is not obvious and to the best of our knowledge neither proposed nor proved in the existing literature that the 3D position of a known sized sphere can be estimated using a single camera image, we propose and prove a bijection principle for it before describing the detection algorithm. The bijection principle, formally stated and proved later in this chapter, proposes that for every given position of a known size spherical object in the 3D world, there is a unique 2D projected image curve. Hence if in an image a curve satisfies the criteria of belonging to the family of spherical object projection curves, it is possible to uniquely identify the spherical object's 3D world position which was projected into that image curve. Exploiting this principle, we present a 3D detection algorithm based on a model fitting approach. This 3D-detector is then plugged into a particle filter-based (PF) tracker to perform a continuous spherical object tracking. A theoretical analysis of the run-time computational complexity for our 3D-detector is presented and compared with

that of a Hough transform (HT) based detector, followed by experimental comparisons for both detectors. An SVS consisting of 2 high resolution cameras is used for the ground truth (GT) evaluation during the experiment. The GT estimates are used to quantify the accuracy of our proposed method.

The rest of the chapter is structured as follows. In Section 2.2 we overview the projection model of the fish-eye lens-based camera used in this work which is essential for the description of the bijective principle detailed in Section 2.3. The details of an HT-based 3D detector and our model fitting approach-based detector are presented in Section 2.4 and 2.5 respectively. This is followed by Section 2.6 where the experimental results are presented including a comparison with the GT. We conclude the chapter with final remarks in Section 2.7.

2.2 Fish-eye Lens Equidistant Projection Model

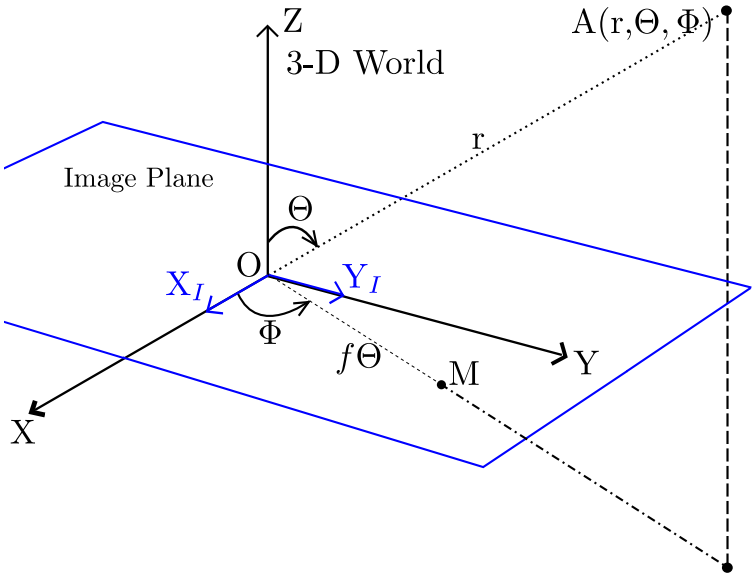


Figure 2.1: 3-D World frame to image frame transformation due to equidistant projection model. The camera lens is facing upwards in the positive Z direction in this figure but it should be noted that when the camera is installed on the robot, the lens faces downwards in the direction of ground plane on which the robot manoeuvres.

In Figure 2.1, A is a point in the 3-D world frame which after being projected through the fisheye lens, falls on the point M which lies on the image plane (blue color plane in Figure 2.1). The projection follows a non-linear transformation function when expressed

in the Cartesian coordinate system, but is linear in the spherical coordinate system (2.1) and is called the equidistant projection model.

$$d = f\Theta, \tag{2.1}$$

where the 3D world frame follows a spherical coordinate system with coordinate variables denoted by r, Θ and Φ and the 2D image frame follows a polar coordinate system with coordinate variables d and Φ. Since Φ remains unchanged after the transformation it is denoted by the same variable. f is the field of view (FOV) constant of the lens computed using the camera's intrinsic parameters.

2.3 The Bijection Principle

In this section we propose and prove the *3D spherical object to 2D image bijection principle* which states that the periphery of a spherical object of known radius when observed through a fish-eye lens which follows the equidistant projection model (2.1), always projects into a unique curve in the image frame for each possible 3D position of that object. Conversely, each curve in the image, which satisfies the condition of being projected from the periphery of a known sized spherical object, back projects into a unique 3D position of that spherical object.

In order to prove this principle, we first introduce the coordinate reference frames for the 3-D spherical object and the image and the transformation from one frame to the other. We then find the expression for the curve (later referred as C_s) of the outer periphery of the sphere in 3D in the object reference frame, which the camera can observe. The projection model (2.1) is then applied to the curve C_s to obtain the projected curve C_i on the image plane in the image coordinate frame. This is followed by the proof of uniqueness of the curve C_i for any 3-D position of the sphere's (initial spherical object which was observed) center and vice-versa hence proving the bijection principle.

2.3.1 Co-ordinates and Transformations

The origin of both the image frame (polar) and the reference plane of the world frame (spherical) is at O in Figure 2.1. The camera's lens which faces upwards in the positive Z axis direction is centred at the origin O in this figure. The Z axis is the zenith direction and the X-Y plane is the reference plane of the world frame as well as the plane which contains the image plane. M is the projected image point of A in the image frame, the

distance of which from O is given by $d = f\Theta$ where f is the FOV constant of the fish-eye lens and Θ (in radians) is the inclination angle, the angle which the ray joining the point $A(r, \Theta, \Phi)$ in 3-D world frame and the lens' optical center (O in Figure 2.1) makes with the optical axis of the lens (Z axis in Figure 2.1). The azimuth angle Φ of the point A in the world frame is equal to the polar angle of its projected point on the image plane. The equidistant projection model (2.1) can also be written as a transformation equation in the matrix form (2.2).

$$\begin{bmatrix} 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r \\ \Theta \\ \Phi \end{bmatrix} = \begin{bmatrix} d \\ \Phi \end{bmatrix} \tag{2.2}$$

The rest of the equations and figures in this proof use these coordinate frames and transformations consistently. (r, Θ, Φ) are henceforth used as the spherical coordinate variables (world frame's coordinate system in which the 3-D object resides) and (d, Φ) are henceforth used as the polar coordinate variables (image frame's coordinate system). Note that the variable Φ is the same in both coordinate systems because the equidistant projection model (2.1) keeps it unchanged and hence a common variable can be used without any loss of generality.

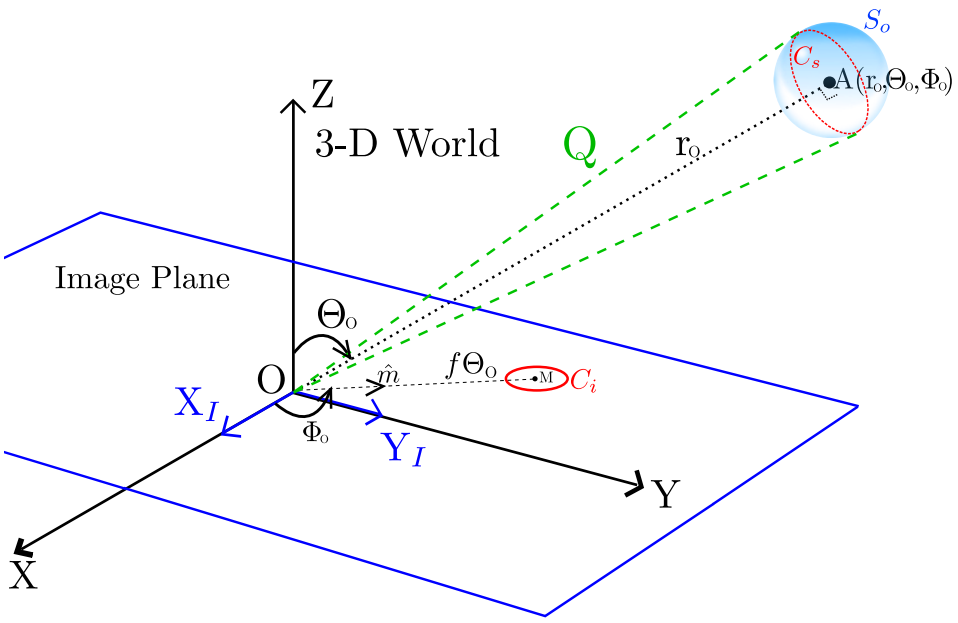


Figure 2.2: 3D object to 2D image projection

2.3.2 Projection Model Applied to a Spherical Object

In Figure 2.2 a spherical object S_o of a known radius R_o is centred at the point A, the coordinates of which are (r_o, Θ_o, Φ_o) in the 3-D world frame. The outer periphery of S_o visible to the camera is the circle C_s which is a circle formed by the contact of S_o's tangential cone Q from O on to S_o itself (see Figure 2.2). The visible periphery C_s is a circle in 3D world frame which forms the base of the cone Q and has a radius R_1 (R_1<R_o) (see Figure 2.3). According to the projection model (2.1), each point on C_s is projected into the image plane forming a curve C_i. In order to prove the proposed bijection principle, it is essential to find the equation of the curve C_i in terms of the polar coordinate variables (d, Φ). The sphere S_o's center's 3-D coordinates: r_o, Θ_o, Φ_o will serve as the parameters for C_i's curve equation. The lens' FOV constant f and the S_o's known radius R_o will be the fixed constants in C_i's curve equation. To do so we first find the equation of C_s in terms of the world frame's spherical coordinate variables (r, Θ, Φ) and then apply (2.1) on the coordinate variables in its equation to find C_i. Next, in order to prove the bijection we will show that for a given triplet of parameters (r_o, Θ_o, Φ_o), *i.e.*, for a given position of the sphere S_o the curve equation C_i represents a unique curve on the image plane. This is followed by proving the converse, *i.e.*, if there exists a curve of the form C_i in the image plane, the coefficients of the terms in C_i's expression will uniquely represent a single point in the world frame's 3D space at which the original spherical object S_o is centred.

2.3.3 Derivation of The Equation of Curve C_s

Observing from Figure 2.2 and 2.3, C_s is a circle which lies on the intersection of the plane PQ (observed as a line in the cross section Figure 2.3) and the sphere S_o. The equation of the the sphere S_o (2.3) in the world reference frame's spherical coordinate system is obtained by transforming a sphere's standard equation in the Cartesian coordinate system into spherical coordinate system.

$$\begin{aligned} S_o(r, \Theta, \Phi) = \; & \{(r, \Theta, \Phi) : r^2 + r_o^2 \\ & -2rr_o(\sin\Theta \sin\Theta_o \cos(\Phi - \Phi_o) \\ & + \cos\Theta \cos\Theta_o) - R_o^2 = 0\} \end{aligned} \tag{2.3}$$

Since the cone Q is tangential to the sphere S_o, all points on the curve C_s are at the same distances from the apex O of the cone Q (see Figure 2.3). As per this argument, in equation (2.3), where r is the coordinate variable denoting the distance of any point on S_o from the origin O, substituting r with the value of l will generate the equation of the curve

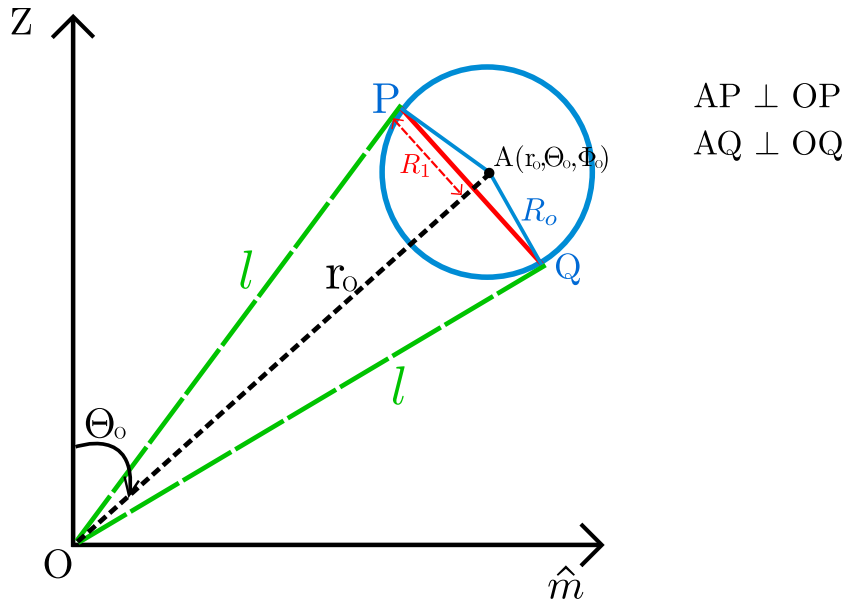


Figure 2.3: Cross-section of Figure 2.2 along the plane perpendicular to X-Y plane and at an azimuth of Φ_0. $\hat{m}$ is the vector along the azimuth of Φ_0 on the reference plane of the world frame in Figure 2.2.

C_s. From Figure 2.3, the value of l in terms of the sphere's parameters and constants is derived as:

$$l = \sqrt{r_o^2 - R_o^2} \tag{2.4}$$

Performing the substitution of r with the value of l in the equation of S_o (2.3), we obtain the equation of the curve C_s as follows:

$$C_s(r, \Theta, \Phi) = S_o(r, \Theta, \Phi)|_{r=l}$$

$$\begin{aligned} \Rightarrow \quad & C_s(r, \Theta, \Phi) = \{(l, \Theta, \Phi) : r_o^2 \\ & -r_o\sqrt{r_o^2 - R_o^2}(\sin\Theta \sin\Theta_o \cos(\Phi - \Phi_o) \\ & + \cos\Theta \cos\Theta_o) - R_o^2 = 0\} \end{aligned} \tag{2.5}$$

2.3.4 Obtaining C_i from C_s

Applying (2.1) on the coordinate variables of the equation of C_s we obtain

$$\begin{aligned} C_i(d,\Phi) = \; & \{(d,\Phi) : r_o^2 \\ & -r_o\sqrt{r_o^2 - R_o^2}(\sin(\tfrac{d}{f})\sin\Theta_o\cos(\Phi-\Phi_o) \\ & +\cos(\tfrac{d}{f})\cos\Theta_o) - R_o^2 = 0\} \end{aligned} \tag{2.6}$$

which is expressed in the image's polar coordinate variables d and Φ, the parameters r_o, Θ_o, Φ_o which are the spherical coordinates of the sphere S_o's center and the fixed constants f (lens' FOV constant) and R_o (sphere S_o's known radius). We represent the curve $C_i(d,\Phi)$ as a function $\mathcal{F}(d,\Phi)$ with constraints on the variables and parameters in the following way:

$$\begin{aligned} \mathcal{F}(d,\Phi) = \{(d,\Phi) : \; & r_o^2 \\ & -r_o\sqrt{r_o^2 - R_o^2}(\sin(\tfrac{d}{f})\sin\Theta_o\cos(\Phi-\Phi_o) \\ & +\cos(\tfrac{d}{f})\cos\Theta_o) - R_o^2 = 0; \\ & 0 < d \le f\tfrac{\pi}{2}; \;\; -\pi \le \Phi \le \pi\} \end{aligned} \tag{2.7}$$

where $R_o, f > 0; r_o > R_o;\; 0 \le \Theta_o \le \frac{\pi}{2};$
$-\pi \le \Phi_o \le \pi$ and $d, \Phi, r_o, \Theta_o, \Phi_o, f, R_o \in \mathbb{R}$

The restricted positive limit of d in (2.7) is due to the fact that the image projection of any real point in the 3-D world frame can attain a maximum value of $d = f\frac{\pi}{2}$ due to the projection model (2.1). The limits of Θ_o in (2.7) are due to the fact that only the points on the positive side of the Z axis can form a real image if the camera's lens is placed at the origin O and is pointing in the positive Z axis direction. The fixed constants: camera lens' FOV constant f and the sphere's radius R_o represent positive valued quantities by definition. r_o, the distance to the center of the sphere from the principal point of the image (which is also the origin of the spherical coordinate system) is greater than the radius of the sphere implying that the camera is always outside the sphere S_o. The rest of the limits on the variables and parameters in (2.7) are simply equivalent to the limits of the coordinate variables in a standard polar or spherical coordinate system.

2.3.5 Proof of Bijection

Principle 1. 3D spherical object to 2D image bijection: *The periphery of a spherical object of known radius, when observed through a fish-eye lens that follows the equidistant projection model (2.1), always projects into a unique curve in the image frame for each possible 3D position of that object. Conversely, each curve in the image, which*

satisfies the condition of being projected from the periphery of a known sized spherical object, back projects into a unique 3D position of that spherical object.

Mathematical Formulation

The bijection principle, as stated above, can be mathematically formulated in the following way: A map $\mathcal{G} : \mathbb{S} \to \mathbb{C}$ is bijective where

$$\begin{aligned}\mathbb{S} = \{(r_o, \Theta_o, \Phi_o) \ : \ r_o > R_o, \ 0 \leq \Theta_o \leq \frac{\pi}{2}, \ -\pi \leq \Phi_o \leq \pi; \\ r_o, \Theta_o, \Phi_o \in \mathbb{R}\},\end{aligned} \tag{2.8}$$

$$\mathbb{C} = \{C_i \ : \ C_i \equiv \mathcal{F}(d, \Phi)\} \tag{2.9}$$

for fixed values of f and R_o.

In order to prove the bijection principle we need to establish the following two statements:

- The map $\mathcal{G} : \mathbb{S} \to \mathbb{C}$ is injective.
- The map $\mathcal{G} : \mathbb{S} \to \mathbb{C}$ is surjective.

Proof of Injection

Simplifying (2.7) we get:

$$\begin{aligned}\mathcal{F}(d, \Phi) \ &= \{(d, \Phi) \ : \ 1 - \alpha \sin(\tfrac{d}{f}) \cos(\Phi) \\ &\quad -\beta \sin(\tfrac{d}{f}) \sin(\Phi) - \gamma \cos(\tfrac{d}{f}) = 0\}\end{aligned} \tag{2.10}$$

where

$$\begin{aligned}\alpha &= \tfrac{r_o \sin\Theta_o \cos\Phi_o}{\sqrt{r_o^2 - R_o^2}}; \\ \beta &= \tfrac{r_o \sin\Theta_o \sin\Phi_o}{\sqrt{r_o^2 - R_o^2}}; \\ \gamma &= \tfrac{r_o \cos\Theta_o}{\sqrt{r_o^2 - R_o^2}}\end{aligned} \tag{2.11}$$

assuming the restrictions on the variables and the parameters to be the same as in (2.7)

In order to prove that the map $\mathcal{G}$ is injective, we need to show that any given triplet (r_o, Θ_o, Φ_o), for all $r_o > R_o,\ 0 \leq \Theta_o \leq \frac{\pi}{2},\ -\pi \leq \Phi_o \leq \pi$, maps to a unique curve of the form C_i which implies that the mapping is one to one. Since a curve is a set of points, it implies that we need to show that a given triplet (r_o, Θ_o, Φ_o) maps to a unique set of points in the polar space of (d, Φ) for all $0 < d \leq f\frac{\pi}{2};\ -\pi \leq \Phi \leq \pi$.

The forward mapping of $\mathcal{G}$ is trivial. Considering the simplified equation (2.10), we can find a set $\mathcal{K}$ of all the points in the polar space (d, Φ) which satisfy (2.10) for a given triplet (r_o, Θ_o, Φ_o). A given triplet (r_o, Θ_o, Φ_o) implies that the values of the coefficients α, β and γ are given.

The mapping $\mathcal{G}$ is one to one, *i.e.*, injective if and only if the set of points $\mathcal{K}$ maps back to the unique triplet (r_o, Θ_o, Φ_o). For the inverse mapping of $\mathcal{G}$, assume that we are given the set of points $\mathcal{K}$ such that all the points $(d, \phi) \in \mathcal{K}$ satisfy the equation of the form (2.10). Let us pick three distinct points $((d_1, \Phi_1), (d_2, \Phi_2), (d_3, \Phi_3)) \in \mathcal{K}$. In order to show that three distinct points exist on the curve (2.10), we will first prove that at least four distinct points exist on (2.10).

Notice that in the family of equations represented by (2.10), the function equated to 0 can be seen in two forms; either Φ as a function of d or d as a function of Φ. By differentiating such representations of this function w.r.t. d in the first form and w.r.t Φ in the other and then separately equating both differentials to zero we can obtain the values for d and Φ at which the function has the maxima and minima. Equations (2.12-2.15) represent these extrema.

$$d_{max} = f(\Theta_o + \arcsin\frac{R_o}{r_o}) \tag{2.12}$$

$$d_{min} = f(\Theta_o - \arcsin\frac{R_o}{r_o}) \tag{2.13}$$

$$\Phi_{max} = \Phi_o + \arcsin\frac{R_o}{r_o \sin\Theta_o} \tag{2.14}$$

$$\Phi_{min} = \Phi_o - \arcsin\frac{R_o}{r_o \sin\Theta_o} \tag{2.15}$$

At the extremum of d, the corresponding value of Φ and at the extremum of Φ, the corresponding value of d can be obtained by plugging the values of the extremum back into (2.10). These points in polar coordinates will be given by $(f(\Theta_o + \arcsin\frac{R_o}{r_o}), \Phi_o)$,

$(f(\Theta_o - \arcsin\frac{R_o}{r_o}), \Phi_0)$, $(f(\arccos\frac{r_o\cos\Theta_o}{\sqrt{r_o^2-R_o^2}}), \Phi_o + \arcsin\frac{R_o}{r_o\sin\Theta_o})$ and $(f(\arccos\frac{r_o\cos\Theta_o}{\sqrt{r_o^2-R_o^2}}), \Phi_o - \arcsin\frac{R_o}{r_o\sin\Theta_o})$. It can now be verified that these represent four distinct points assuming the constraints on the constants of (2.10) explained earlier in this section.

Proceeding with the proof of injection, since by initial assumption all the points in the set $\mathcal{K}$ satisfy the curve equation (2.10), the following three equations are true:

$$\alpha\sin(\frac{d_1}{f})\cos(\Phi_1) + \beta\sin(\frac{d_1}{f})\sin(\Phi_1) + \gamma\cos(\frac{d_1}{f}) = 1 \tag{2.16}$$

$$\alpha\sin(\frac{d_2}{f})\cos(\Phi_2) + \beta\sin(\frac{d_2}{f})\sin(\Phi_2) + \gamma\cos(\frac{d_2}{f}) = 1 \tag{2.17}$$

$$\alpha\sin(\frac{d_3}{f})\cos(\Phi_3) + \beta\sin(\frac{d_3}{f})\sin(\Phi_3) + \gamma\cos(\frac{d_3}{f}) = 1 \tag{2.18}$$

In the case of inverse mapping of $\mathcal{G}$, $\mathcal{K}$ is given and hence known but the triplet (r_o, Θ_o, Φ_o) is unknown, hence α, β, γ become variables. Equations (2.16) to (2.18) can now be seen as a system of 3 equations in 3 variables: α, β, γ. Expressing this system in matrix form:

$$\mathcal{A}\rho = 1_3 \tag{2.19}$$

where

$$\mathcal{A} = \begin{bmatrix} \sin(\frac{d_1}{f})\cos(\Phi_1) & \sin(\frac{d_1}{f})\sin(\Phi_1) & \cos(\frac{d_1}{f}) \\ \sin(\frac{d_2}{f})\cos(\Phi_2) & \sin(\frac{d_2}{f})\sin(\Phi_2) & \cos(\frac{d_2}{f}) \\ \sin(\frac{d_3}{f})\cos(\Phi_3) & \sin(\frac{d_3}{f})\sin(\Phi_3) & \cos(\frac{d_3}{f}) \end{bmatrix}, \tag{2.20}$$

$\rho = \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix}^{\mathrm{T}}$ and $1_3 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}$.

The system in (2.19) has a unique solution for ρ if and only if $\mathcal{A}$ is invertible. Also, from (2.11) it can be easily verified that the triplet (r_o, Θ_o, Φ_0) can be uniquely obtained from the triplet (α, β, γ). Hence in order to show that the map $\mathcal{G}^{-1}$ will map from a set $\mathcal{K}$ to a unique triplet (r_o, Θ_o, Φ_0) and therefore prove that the map $\mathcal{G}$ is injective, it will suffice to show that $\mathcal{A}$ is invertible under the initial assumptions of the problem.

In the expression of $\mathcal{A}^{-1}$ (2.21), the inverse of matrix $\mathcal{A}$, it can be verified that under the previously mentioned variables and constants limits, all elements of $\mathcal{A}^{-1}$ will have a finite value thus proving its existence. Hence the map $\mathcal{G}$ is injective.

Proof of Surjection

The proof of surjection is now trivial. A function is surjective if its image is equal to its codomain. We have already shown that any set of points $\mathcal{K} = \{(d, \Phi)\}$ in the polar

$$\mathcal{A}^{-1} = \begin{bmatrix} \frac{\frac{\sin(\frac{d_3}{f})\sin(\phi_1)}{\tan(\frac{d_2}{f})} - \cos(\frac{d_3}{f})\sin(\Phi_2)}{\sin(\frac{d_1-d_3}{f})\sin(\Phi_1-\Phi_2)} & \frac{\sin(\phi_1)}{\sin(\frac{d_2}{f})\sin(\Phi_1-\Phi_2)} & \frac{-\frac{\sin(\frac{d_1}{f})\sin(\phi_1)}{\tan(\frac{d_2}{f})} + \cos(\frac{d_1}{f})\sin(\Phi_2)}{\sin(\frac{d_1-d_3}{f})\sin(\Phi_1-\Phi_2)} \\ \frac{\frac{-\sin(\frac{d_3}{f})\cos(\phi_1)}{\tan(\frac{d_2}{f})} + \cos(\frac{d_3}{f})\cos(\Phi_2)}{\sin(\frac{d_1-d_3}{f})\sin(\Phi_1-\Phi_2)} & -\frac{\cos(\phi_1)}{\sin(\frac{d_2}{f})\sin(\Phi_1-\Phi_2)} & \frac{\frac{\sin(\frac{d_1}{f})\cos(\phi_1)}{\tan(\frac{d_2}{f})} - \cos(\frac{d_1}{f})\cos(\Phi_2)}{\sin(\frac{d_1-d_3}{f})\sin(\Phi_1-\Phi_2)} \\ -\frac{\sin(\frac{d_3}{f})}{\sin(\frac{d_1-d_3}{f})} & 0 & \frac{\sin(\frac{d_1}{f})}{\sin(\frac{d_1-d_3}{f})} \end{bmatrix} \tag{2.21}$$

coordinate space which satisfies the equation of the form (2.10) (i.e., $\mathcal{K}$ is in the codomain $\mathbb{C}$ of the mapping $\mathcal{G}$) maps back to a unique triplet $(r_o, \Theta_o, \Phi_o) \in \mathbb{S}$ implying that $\mathcal{K}$ is also in the range of $\mathcal{G}$. Therefore the codomain of the map $\mathcal{G}$ is equal to its image. Hence the map is surjective.

A mapping is bijective if and only if it is injective and surjective at the same time. By establishing that the map $\mathcal{G}$ is injective and surjective, we conclude that it is bijective. Hence proving the proposed bijection principle, Q.E.D.

Concluding from this and the previous sub-sections in this section, it is possible to uniquely identify the position of a known dimension spherical object in 3D using a single equidistant projected image. Using the proposed principle and its proof, in Section 2.5 we will describe a fast algorithm to detect curves of the form C_i (henceforth referred to as tear-drop curves because of their visual appearance) in the image and then find the parameters of those. The 3D position of the center of the spherical object can be found using these parameters in a straight forward manner.

In order to experimentally validate the correctness of the tear drop curve's function, derived mathematically in this section, we implemented a Hough transform-based (HT) curve detector for curves of the form C_i on a synthetic image generated for few known 3D positions of a spherical object. We also evaluate, if such a detector is feasible for real-time tear drop curve detection. This is presented in Section 2.4.

2.4 Hough Transform (HT) for Tear-drop Curves

HT is a very well known tool for curve or shape detection [40]. An HT-based C_i-type curve detector was applied on a synthetically generated binary image (see, e.g, Figure 2.4) which contains the tear drop curves. The results show that the parameters found using such a

detector, correctly correspond to the 3D positions of the sphere's center (those initially used to generate the synthetic image).

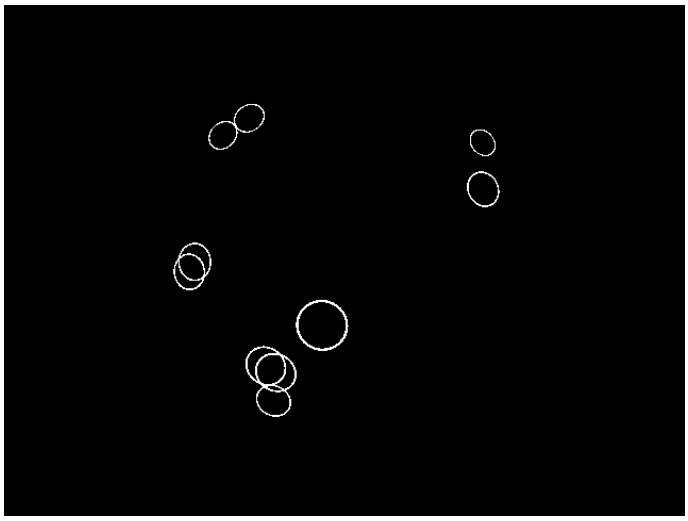


Figure 2.4: Binary edge image containing the projected tear-drop curves for a set of random 3D positions of a known-diameter sphere.

2.4.1 Computation Time Analysis of The HT-based Detector

To verify if an HT-based approach was suitable for real-time detection of tear drop curves, a computation time analysis was done. Consider the HT-based detector applied on an image containing N pixels out of which N_e pixels are considered to be edge points. If the number of parameters describing the curve to be detected is p and the implementation assumes n discretization levels for each parameter for the construction of the accumulator bin, then the run-time complexity for the worst-case scenario of the HT-based detector is $O(N_e n^p)$. In the tear-drop case the parameters are r, Θ and Φ, hence $p = 3$ therefore the run-time complexity will be $O(N_e n^3)$, i.e., cubic time with respect to the number of discretization levels for each parameter and linear time with respect to detected edge points.

To experimentally find the computational speed of the HT-based detection, a synthetic image with only one tear-drop curve was constructed and subjected to the HT-based detection. This was implemented on a quad-core Intel(R) Core(TM) i3 CPU M 350 @ 2.27GHz with a 2.9 GiB of RAM. The time taken to perform the detection was 0.43 seconds when one of the coordinates of the sphere's center was fixed. This clearly highlights the inefficiency

of using the HT-based detection in real-time. Without fixing any of the sphere's center's coordinate, computation time would further increase. It is worth mentioning that the HT-based detector works on an an edge image, meaning that an edge detection algorithm must be applied on the real camera images which will further slow down the detector.

2.5 Model-Fitting Approach

From the previous section we conclude that an HT-based detector for tear-drop curves is not suitable for detecting a sphere in 3D space in real-time. Hence a new detector is created based on a model-fitting approach to solve this problem. This approach exploits the geometrical structure and the curve equation (2.6) of the tear-drop in an efficient way which is explained in the further subsections. It should be noted that in this approach the need for edge detection is removed since it works directly on a color-segmented binary image. This further increases the detection speed.

2.5.1 Algorithm

Algorithm 2.1 Model_Fitting_Algorithm(I_o)

1: Apply color segmentation on I_o and obtain image I_{seg}.
2: Apply erosion and dilation on I_{seg}.
3: Find connected components in I_{seg} and mark them as blobs.
4: Find and fit convex hulls on each blob present in I_{seg}.
5: Find the radial and azimuthal extremities of each convex hull found in Step 4.
6: Use the extremities, found in Step 5, in (2.12-2.15) to solve for r_o, Θ_o and Φ_o for each convex hull to classify blobs as a tear-drop.
7: Calculate mismatch factor for each tear-drop.
8: Choose the tear-drop with the least mismatch factor below a pre-defined threshold and return r_o, Θ_o and Φ_o corresponding to it.

The model fitting approach algorithm is presented in Alg. 2.1. After grabbing an image frame, color-segmentation is applied on it to generate a binary image where pixels corresponding to the sphere's known color are rendered white and the rest are rendered as black. Subsequently, erosion and dilation are performed on the segmented binary image in order to remove spurious noise and to obtain distinct blobs in it by finding connected components (or contours). Each such blob obtained is a candidate for the projection of

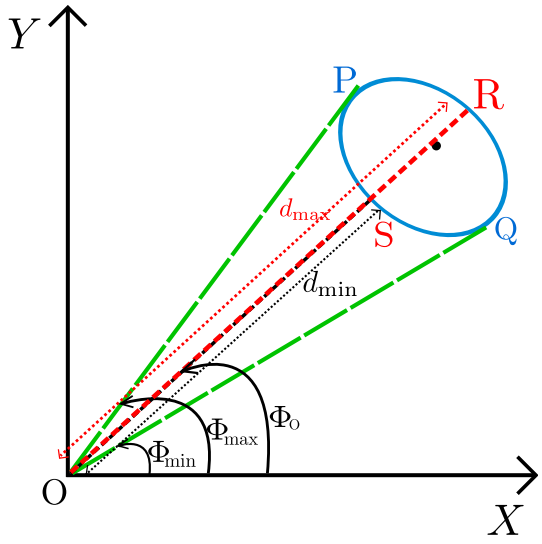


Figure 2.5: The curve PRQS in the figure is an illustration of the tear-drop curve which forms the outer periphery of a candidate blob in the image plane (note that it is not an ellipse although it appears to be so in this synthetic figure). Φ and d are the polar coordinate representations.

the sphere in 3D onto the image. Note here that the tear-drop curve (2.6) is the equation of the outer periphery of the projection. This is the reason why in the HT-based detection we needed the edges of these blobs to perform the detection. However, in this approach we do not need the edges exclusively. In the next step a convex hull is fit around each candidate blob. All further computations are performed on these convex hulls.

Before proceeding with the explanation of the next step of the algorithm, it is necessary to explain the geometrical properties of the tear-drop which are exploited using the convex hulls obtained so far. In Figure 2.5 a tear-drop (curve PRQS) is illustrated which represents the periphery of a candidate blob obtained in the step 3 of Alg. 2.1. The tear-drop equation (2.6) assumes the center of the image to be the origin of the polar coordinates. Using (2.6) we can find the extremities of it in terms of the curve's parameters r_o, Θ_o and Φ_o. These parameters eventually denote the sphere's center in 3D space. The extremities are the points on the tear-drop curve having the maximum and minimum values of the d and Φ coordinates. In Figure 2.5 they are R, S, P and Q respectively. Equations (2.12-2.15) express these extremities in terms of r_o, Θ_o, Φ_o, f, the lens' known FOV constant and R_o, the known radius of the spherical object. This is done by differentiating the function in (2.10) with respect to d and Φ and then separately equating both differentials to zero, as explained in sub-section 2.3.5 when proving the proposed bijection principle.

Further in Alg. 2.1, the extreme points in d and Φ for each convex hull is computed. If the blob enclosed by a convex hull is the actual projection of the sphere on to the image, then its convex hull's extremities should be equal to the tear-drop curve's extremities. This is the property which we exploit here. The extreme points of the convex hull are plugged in the equations (2.12-2.15) in order to have a system of 4 equations in 3 variables. Solving any 3 of these equations and checking for redundancy using the extra fourth equation leads us to characterize the blob either as a projected tear-drop curve or as a random blob obtained from some other object of similar color.

The mismatch factor of the blob's classification as a tear-drop curve is influenced by the ratio of the number of pixels in the blob to the area of the convex hull as well as the distance to the blob from the image center. Experimentally finding practical thresholds for the mismatch factor, spuriously classified tear-drop curves are eliminated from classification in step 6 of Alg. 2.1.

The convex hull, which is classified as a proper tear-drop curve's convex hull as per step 6 and has the minimum mismatch factor as per step 7 of Alg. 2.1, is classified as the actual projection of the sphere in 3D space. Again using the equations (2.12-2.15) for the extremities of this convex hull, the parameters r_o, Θ_o and Φ_o are obtained which was the desired result of the detection algorithm.

2.5.2 Computation Time Analysis

Consider the model fitting algorithm applied on an image containing N pixels. The color segmentation, erosion and dilation steps, each have a run-time complexity for the worst-case scenario of $O(N)$. The step 3 of Alg.2.1 uses a method for finding connected components to identify distinct blobs in a binary image which was proposed by Suzuki and Abe in [41]. This method's worst-case scenario run-time complexity is also $O(N)$. For fitting a convex hull around each blob, a method from Sklansky [42] is used which has a worst-case run-time complexity of $O(bN_v)$ where b is the number of blobs in the image and N_v is the average number of vertex points of each blob. Theoretically the maximum possible value of the product bN_v is N. The equation solving, likelihood assigning and finally searching for maximum likely tear-drop steps in Alg.2.1, each have a worst-case run-time complexity of $O(b)$. Summing all of them and considering the highest order term while dropping the constants, the worst-case run-time complexity of the model fitting algorithm is $O(N)$ where N is the total number of pixels in the image. This shows that the proposed algorithm is clearly much faster compared to the HT-based approach which not only has an additional edge detection step with the run-time complexity of $O(N)$ but also the rest

of the algorithm has a complexity of cubic time in the number of discretization levels of the curve parameters. In the model fitting algorithm there is no such restriction of discretizing the parameters.

The model fitting algorithm was implemented on the same machine as the one used for the HT-based approach and mentioned in the previous sub-section. It took $\sim$ 38 milliseconds to run the whole algorithm (Alg. 2.1) on a real image frame compared to the HT-based approach which took 430 milliseconds to perform only the detection of a single tear-drop curve on a synthetic image (Note that a synthetic image has zero noise so there is no need for erosion or dilation for it). This makes the model fitting algorithm at least 11 times faster than the HT-based approach. With this computation time, we were able to perform detection at $\sim$ 26 image frames per second. It is evident that the HT-based approach will be even slower if it was implemented on noisy real images after performing color segmentation and edge detection on them before proceeding to the detection step.

2.6 Experimental Setup and Results

A particle filter-based (PF) tracker was constructed which involved the classical PF's predictor and resampler along with the model fitting approach as the classifier, introduced as one of the novel contributions of this thesis. The prediction and the resampling steps inherit directly from the original PF [43]. The model fitting approach-based detector is used for assigning weights to the particles in the update step.

We implemented the tracker in the robot soccer scenario (see Appendix A for a detailed description of the testbed). One of the primary requisites for any robot to be able to play soccer is to continuously track the soccer ball. In robotic soccer's dynamic environment the ball maneuvers in a 3D space which makes the robot soccer scenario an interesting and suitable test bed for the experimental validation and performance check of our approach. The robot tracks an orange colored ball (standard FIFA size 5) and move towards it while the ball is manually moved around in the 3D space.

In order to evaluate the results of our approach's implementation, we compare the results with the GT (see Appendix B for the GT system's details). The results presented in this chapter is of the experiment performed in the LSA lab facility of ISEP Porto, Portugal, where a GT system was already installed. Later a similar GT system was installed in ISR/IST and used for experiments presented in the later chapters of this thesis. Figure 2.6 shows one of the frames of the video footage from the GT system at ISEP, Porto.

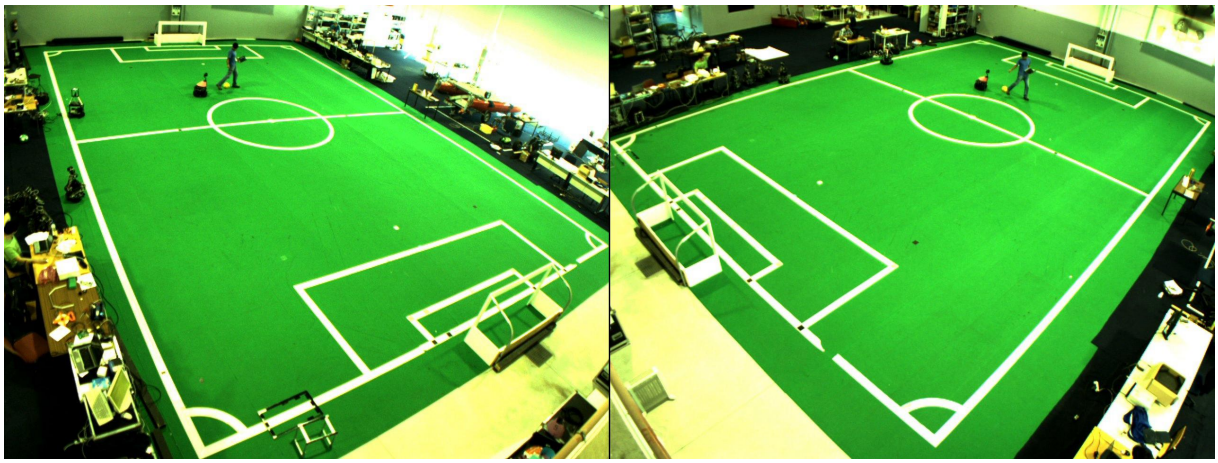


Figure 2.6: Snapshot from the stereo vision system installed for GT evaluation of the robot and the ball's 3D positions. Result presented in this chapter is of the experiment performed in the LSA lab facility of ISEP Porto, Portugal.

2.6.1 Results

Figure 2.7 shows the comparison of the Z-coordinate of the tracked 3D position of the ball against the ground truth for approximately 570 iterations of the PF. On an average, each iteration of the PF takes around 0.1 seconds hence the experiment presented here spans a duration of about 1 minute.

In Figure 2.8 the 2D positions are plotted for the estimated ball position by the robot, robot's self-localization estimate and the ground truth positions of both as estimated by the GT system. It should be noted that the GT system detects only the 2D ground positions of the robot and the 3D positions of ball but not the orientation of the robot, hence a direct X or Y coordinate comparison of the estimated ball position by the robot against its ground truth is not possible. However, a comparison of the estimated radial distance of the ball from the robot with the distance between the GT positions for both can be done. In Figure 2.9 and 2.10 the plots present the error values computed as the absolute value of the Euclidean distance between the estimated and the GT positions. The mean and variance of these errors are presented in Table 2.1.

2.7 Summary

In this chapter we proposed and proved a bijection principle which states that the 3D position of a spherical object of known diameter, when projected onto an image frame of

	Z-coordinate	**Radial distance of the ball from the robot**
mean error (m)	0.049	0.093
variance of error (m^2)	0.001	0.013

Table 2.1: Results of the 3D tracking using the model fitting approach as the 3D spherical object detector.

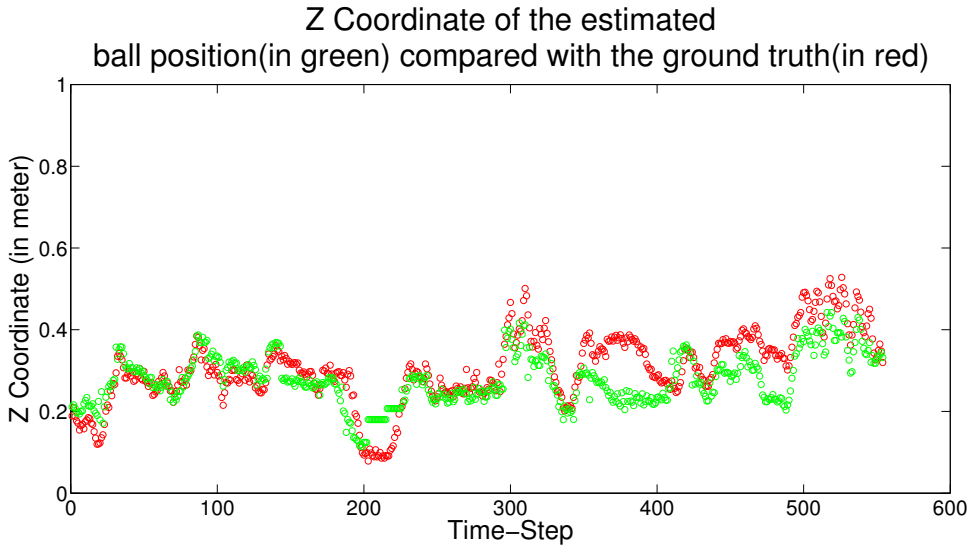


Figure 2.7: Comparison of the tracked position of the ball's Z coordinate (green) against the ground truth's Z coordinate (red). X-axis in the figure represents PF iteration. Y-axis in figure represents the Z-coordinate of the tracked ball positions and the corresponding ground truth.

a fish-eye lens-based camera, can be uniquely identified using only a single image frame. We presented a detailed mathematical proof for this principle and then later used it to construct a fast algorithm for the 3D detection of a spherical object of known diameter. We also gave the run-time computational complexity for it and compared with an HT-based approach. This detector was implemented as a classifier in a PF-based tracker to perform continuous tracking of a FIFA standard size 5 ball by a mobile robot installed with a fish-eye lens-based camera. The results of the tracking were compared against the GT obtained using a stereo vision system consisting of high resolution ethernet cameras. We achieved a high degree of accuracy using our proposed method, supported by the mean error of $\sim$ 0.05m in the Z coordinate, $\sim$ 0.09m in the radial distance to the ball from

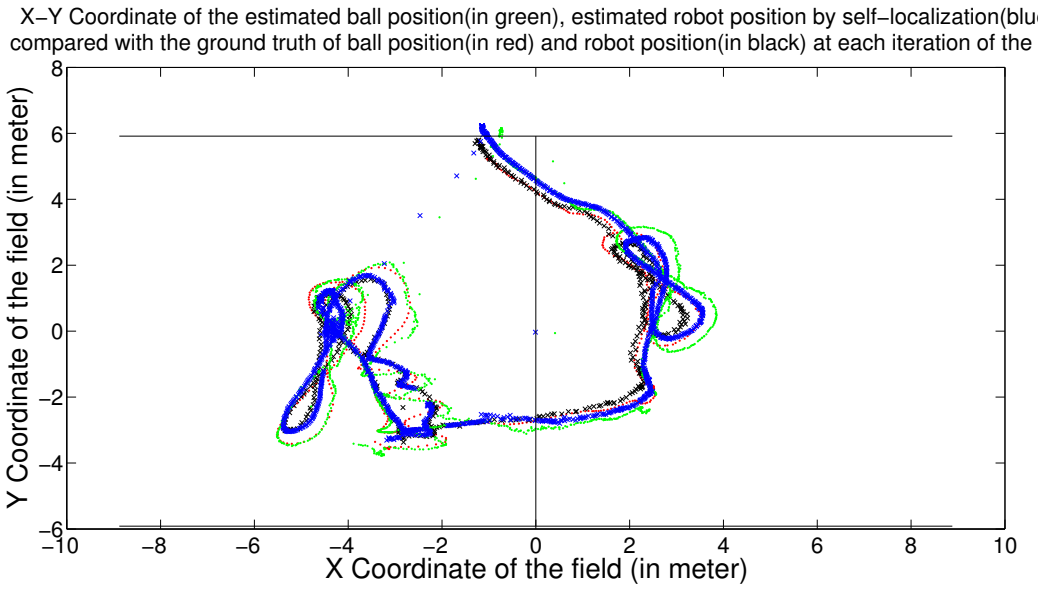


Figure 2.8: Comparison of the tracked position of the ball and the robot's XY coordinate (green,blue) against the ground truth's XY coordinates for the same (red,black). The X and Y-axis in the figure respectively denote the X and Y coordinates of the ball's tracked and ground truth positions.

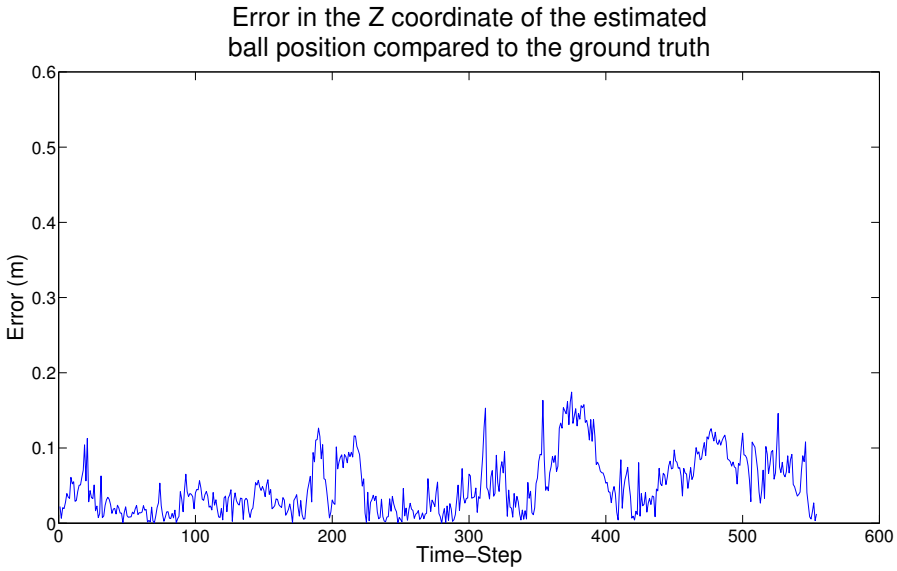


Figure 2.9: Error in the estimated ball position's Z coordinate at each step of the PF iteration.

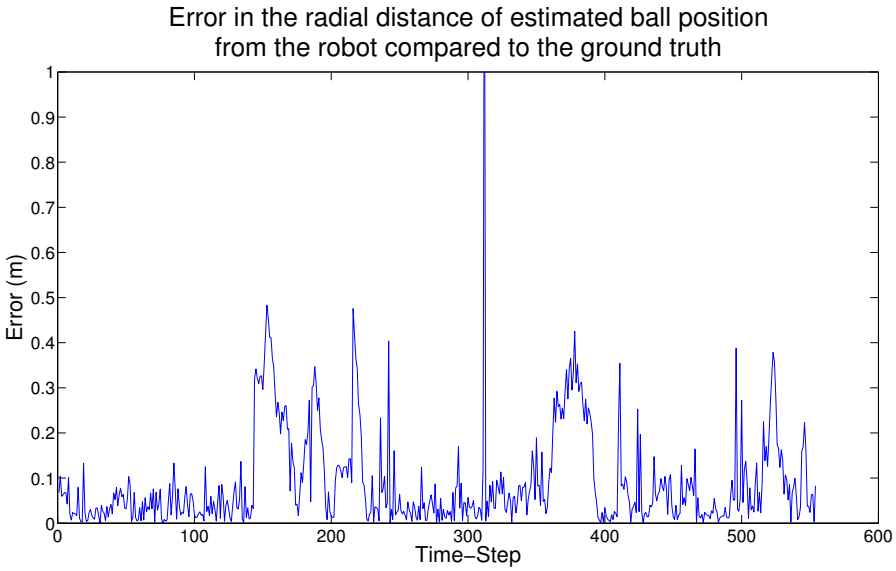


Figure 2.10: Error in the estimated radial distance of the ball position from the robot at each step of the PF iteration.

the tracking robot and variance in both $\sim 0.0014\text{m}^2$. Ongoing work includes developing algorithms to remove the need of color in the detection process and to include generic objects with more complex 3D geometry.

Chapter 3

Multi-Robot Cooperative Object Tracking

3.1 Introduction

In this chapter we introduce an approach to multi-robot cooperative object tracking (MCOT) through a particle filter-based (PF) tracker [9]. For each observing robot (i.e., a mobile robot with a sensor), we determine confidence factors associated to the tracked target from two origins: i) the confidence on the observation measurement itself and ii) the confidence on the self-localization estimate of the observing robot. Each robot's self-localization method itself is completely decoupled from the PF-based cooperative object tracker running on it. The observation measurements obtained by the robots' sensors are parametrized (e.g., a Gaussian with a mean and a covariance). The parametrized observation measurements of the team robots are shared by all of them in a pool. At each robot, the PF-based cooperative tracker selects, for a particle, a measurement from the pool with a probability proportional to the measurement's confidence and uses it to assign weight to that particle. Repeating this for every particle, the filter eventually fuses all the measurements in the pool. The parametrization also intends to reduce significantly the amount of data communicated among the teammates. This approach to MCOT handles, within a common framework, inconsistencies (disagreements) between sensors due to observation measurement errors and/or robots' self-localization uncertainties.

The rest of the chapter is organized as follows. In Section 3.2 we introduce a standard individual PF-based object tracker. In Section 3.3 we introduce the fusion step and explain how we incorporate it in a standard PF to construct the PF-based cooperative tracker, the core of this chapter. Section 3.4 presents the proposed method's implementation details

and real robots experiment results compared with the ground truth. A summary with comments on the results is presented in Section 3.5.

3.2 A standard Particle Filter-based Tracker

Let us consider a team of N robots, $\mathrm{r}_1, \ldots, \mathrm{r}_N$. Robot r_i has pose (position + orientation) coordinates $\mathcal{L}_t^{\mathrm{r}_i} = [x_t^{\mathrm{r}_i} \quad y_t^{\mathrm{r}_i} \quad \theta_t^{\mathrm{r}_i}]^\top$ in a global world frame at the t^{th} timestep (in this chapter we implicitly assume to know the robots' poses obtained through a self-localization algorithm running separately from the object tracker on the same robots[1]).

In this section we briefly introduce a standard PF-based object tracker running on a robot r_i and tracking the object $\mathbf{O}$ without cooperating with any other robot in the team. The state vector corresponding to the 3D global-frame position of an object at the t^{th} timestep, which our tracker running on the robot r_i estimates, is denoted by $\mathbf{O}_t^{\mathrm{r}_i} = [x_t^{\mathbf{o},\mathrm{r}_i} \; y_t^{\mathbf{o},\mathrm{r}_i} \; z_t^{\mathbf{o},\mathrm{r}_i}]^\top$. We assume that the object's state evolution over time is a Markov process with a uniform initial distribution $p(\mathbf{O}_0^{\mathrm{r}_i})$ and a transition distribution $p(\mathbf{O}_t^{\mathrm{r}_i}|\mathbf{O}_{t-1}^{\mathrm{r}_i}, \mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}})$, where $\mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}}$ is the velocity input to the tracked object's motion model. The observation measurements $\{\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}; t \in \mathbb{N}\}$, which are the detected 3D positions of the tracked object by the robot r_i, are conditionally independent given the process $\{\mathbf{O}_t^{\mathrm{r}_i}; t \in \mathbb{N}\}$ with distribution $p(\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}|\mathbf{O}_t^{\mathrm{r}_i})$.

The posterior belief $bel(\mathbf{O}_t^{\mathrm{r}_i}) = p(\mathbf{O}_t^{\mathrm{r}_i}|\mathbf{z}_{1:t}^{\mathrm{r}_i,\mathbf{o}}, \mathbf{v}_{1:t}^{\mathrm{r}_i,\mathbf{o}})$ is a probability distribution of the posterior at the t^{th} timestep over the state space, given all the control (object's velocity in this case) and the observation measurements up to that timestep. To estimate this posterior distribution, we use a recursive Bayesian filter, with the Markovian assumption of the state's completeness, formulated as:

$$p(\mathbf{O}_t^{\mathrm{r}_i}|\mathbf{z}_{1:t}^{\mathrm{r}_i,\mathbf{o}}, \mathbf{v}_{1:t}^{\mathrm{r}_i,\mathbf{o}}) \propto p(\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}|\mathbf{O}_t^{\mathrm{r}_i}) \int p(\mathbf{O}_t^{\mathrm{r}_i}|\mathbf{O}_{t-1}^{\mathrm{r}_i}, \mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}}) p(\mathbf{O}_{t-1}^{\mathrm{r}_i}|\mathbf{z}_{1:t-1}^{\mathrm{r}_i,\mathbf{o}}, \mathbf{v}_{1:t-1}^{\mathrm{r}_i,\mathbf{o}}) d\mathbf{O}_{t-1}^{\mathrm{r}_i} \tag{3.1}$$

In the rest of the chapter we refer to $p(\mathbf{O}_t^{\mathrm{r}_i}|\mathbf{O}_{t-1}^{\mathrm{r}_i}, \mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}})$ as the *tracked object's motion model* and $p(\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}|\mathbf{O}_t^{\mathrm{r}_i})$ as the *tracked object's observation model*.

For solving the problem as formulated above we use a PF-based tracker on the robot r_i. A PF is a non-parametric Bayesian filter where, contrary to a parametric Bayesian filter, the state's posterior probability distribution is represented by a set of M particles $\mathcal{X}_t^{\mathbf{o}} \triangleq \{\langle \mathbf{x}_t^{[m],\mathbf{o}}, w_t^{[m],\mathbf{o}} \rangle\}_{m=1}^M$. The components of any m^{th} particle $\langle \mathbf{x}_t^{[m],\mathbf{o}}, w_t^{[m],\mathbf{o}} \rangle$ are $\mathbf{x}_t^{[m],\mathbf{o}}$, a hypothesis (and a concrete instantiation) of the state $\mathbf{O}_t^{\mathrm{r}_i}$ and an associated weight

[1]The robot localization algorithm (cooperative and non-cooperative) is discussed in Chapter 4

$w_t^{[m],\mathbf{o}}$ which, ideally, should be proportional to its Bayes filter posterior $bel(\mathbf{O}_t^{\mathrm{r}_i})$ [43]. A standard PF-based object tracker consists of a prediction-update-resample loop, where the prediction of all the particles' state hypothesis component is performed by the following sampling equation:

$$\bar{\mathbf{x}}_t^{[m],\mathbf{o}} \sim p(\mathbf{O}_t^{\mathrm{r}_i} | \mathbf{x}_{t-1}^{[m],\mathbf{o}}, \mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}}), \tag{3.2}$$

using the particle set $\mathcal{X}_{t-1}^{\mathbf{o}}$ resulting from an immediate previous iteration of the standard PF-based tracker and the most recent control input $\mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}}$. The update step computes the weight component $\bar{w}_t^{[m],\mathbf{o}}$ of all the temporary predicted particles, subsequently stored in a temporary particle set $\bar{\mathcal{X}}_t^{\mathbf{o}} \triangleq \{\langle \bar{\mathbf{x}}_t^{[m],\mathbf{o}}, \bar{w}_t^{[m],\mathbf{o}} \rangle\}_{m=1}^{M}$. The weight component is computed by

$$\bar{w}_t^{[m],\mathbf{o}} \propto p(\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}} | \bar{\mathbf{x}}_t^{[m],\mathbf{o}}), \tag{3.3}$$

where the most recent observation measurement $\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}$ is incorporated. Eventually the resampling step selectively draws with replacement particles from the temporary set $\bar{\mathcal{X}}_t^{\mathbf{o}}$ with probability proportional to the weight components of those particles and stores them in the new particle set $\mathcal{X}_t^{\mathbf{o}}$, the final output of the standard PF-based tracker.

3.3 Particle Filter-based Cooperative Tracker

In this section we present our PF-based cooperative tracker algorithm (see Algorithm 3.1). It involves the original (standard) PF algorithm [43], explained in the previous section, augmented with the fusion step which we introduce as a novel contribution. Following the notations introduced in the previous section, here the superscript r_i is modified to denote variables corresponding to other robots in the team.

The prediction and the resampling steps of the PF-based cooperative tracker (lines 2–4 and line 26 of the Algorithm 3.1, respectively) inherit directly from the original PF [43]. We introduce a *fusion step* (lines 5–25) which modifies the particles' weight generation process in a way such that the tracked object's observation measurement made by the robot running the Algorithm 3.1 is fused with those made by the teammate robots.

For a clear description of the algorithm, r_i denotes the robot on which the Algorithm 3.1 is running. The first input to the Algorithm 3.1 is $\mathcal{X}_{t-1}^{\mathbf{o}} \triangleq \{\langle \mathbf{x}_{t-1}^{[m],\mathbf{o}}, w_{t-1}^{[m],\mathbf{o}} \rangle\}_{m=1}^{M}$. It is the set of particles resulting from the immediate previous iteration of the algorithm itself. At the tracker's first timestep initialization, $\mathcal{X}_{t-1}^{\mathbf{o}}$ could be distributed on the state space according to any choice, e.g, a uniform distribution. $\mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}}$ denotes the velocity input to the motion model of the tracked object. M is the total number of particles, the value of which depends on the available computational resource. The larger the value of M, the

Algorithm 3.1 PF_Cooperative_Tracker($\mathcal{X}_{t-1}^{\mathbf{o}}, \mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}}, M, \mathrm{r}_i, N$)

1: $\bar{\mathcal{X}}_t^{\mathbf{o}} \triangleq \{\langle \bar{\mathbf{x}}_t^{[m],\mathbf{o}}, \bar{w}_t^{[m],\mathbf{o}} \rangle\}_{m=1}^{M} = \text{NULL}$
2: **for** $m = 1$ to M **do**
3: $\quad \bar{\mathbf{x}}_t^{[m],\mathbf{o}} = \textbf{\textit{sample\_object\_motion\_model}}\ (\mathbf{x}_{t-1}^{[m],\mathbf{o}}, \mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}})$
4: **end for**
5: {*The Fusion step starts here*}
6: Perform sensor observation and generate $\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}$ and $\Sigma_t^{\mathrm{r}_i,\mathbf{o}}$, where $\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}$ is the object's measurement vector in the world frame and $\Sigma_t^{\mathrm{r}_i,\mathbf{o}}$ is the noise covariance matrix associated with this measurement.
7: Compute object observation confidence $\mathcal{C}_t^{\mathrm{r}_i,\mathbf{o}}$, self-localization confidence $\mathcal{C}_t^{\mathrm{r}_i}$ and Send $\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}$, $\Sigma_t^{\mathrm{r}_i,\mathbf{o}}$, $\mathcal{C}_t^{\mathrm{r}_i,\mathbf{o}}$,$\mathcal{C}_t^{\mathrm{r}_i}$ to the teammate robots.
8: **for** $n = 1$ to N **do**
9: $\quad$ **if** $n == i$ **then**
10: $\quad\quad \alpha^{\mathrm{r}_n} = \mathcal{C}_t^{\mathrm{r}_n,\mathbf{o}}$
11: $\quad$ **else**
12: $\quad\quad$ receive $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$, $\Sigma_t^{\mathrm{r}_n,\mathbf{o}}$, $\mathcal{C}_t^{\mathrm{r}_n,\mathbf{o}}$,$\mathcal{C}_t^{\mathrm{r}_n}$ from the teammate robot r_n
13: $\quad\quad \alpha^{\mathrm{r}_n} = \mathcal{C}_t^{\mathrm{r}_n}\ \mathcal{C}_t^{\mathrm{r}_n,\mathbf{o}}$
14: $\quad$ **end if**
15: **end for**
16: **for** $n = 1$ to N **do**
17: $\quad \alpha^{\mathrm{r}_n} = \frac{\alpha^{\mathrm{r}_n}}{\sum_{n=1}^{N} \alpha^{\mathrm{r}_n}}$ {Normalization step}
18: **end for**
19: **for** $m = 1$ to M **do**
20: $\quad$ draw r_k from $[\mathrm{r}_1, ..., \mathrm{r}_N]$ with probability α^{r_k}
21: $\quad \bar{w}_t^{[m],\mathbf{o}} \propto p(\mathbf{z}_t^{\mathrm{r}_k,\mathbf{o}} \mid \bar{\mathbf{x}}_t^{[m],\mathbf{o}})$
22: $\quad \bar{\mathcal{X}}_t^{\mathbf{o}} = \bar{\mathcal{X}}_t^{\mathbf{o}} + \langle \bar{\mathbf{x}}_t^{[m],\mathbf{o}}, \bar{w}_t^{[m],\mathbf{o}} \rangle$
23: **end for**
24: {*The Fusion step ends here*}
25: $\mathcal{X}_t^{\mathbf{o}} = \textbf{\textit{Resample}}\ (\bar{\mathcal{X}}_t^{\mathbf{o}})$
26: return $\mathcal{X}_t^{\mathbf{o}}$

better is the approximation of the target's *a posteriori* distribution. The algorithm is fully computationally decentralized, meaning that each robot r_n in $[\mathrm{r}_1, ..., \mathrm{r}_N]$ where N is the total number of robots in the team, will run its own instance of the algorithm. We assume that each robot can communicate with every other robot in the team.

In line 1 of the Algorithm 3.1, a temporary particle set $\bar{\mathcal{X}}_t^{\mathbf{o}} \triangleq \{\langle \bar{\mathbf{x}}_t^{[m],\mathbf{o}}, \bar{w}_t^{[m],\mathbf{o}} \rangle\}_{m=1}^{M}$ is initialized with null values. In lines 2–4, the prediction step is performed, where the particles from the set $\mathcal{X}_{t-1}^{\mathbf{o}}$ are propagated based on the tracked object's motion model and the velocity input $\mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}}$ to it and stored in $\bar{\mathcal{X}}_t^{\mathbf{o}}$. The velocity should be calculated using a separate object velocity sensor, e.g, a velocity sensor based on a linear regression of a certain number of previously estimated object's position.

In line 6, the robot r_i performs the object observation measurement in the global reference frame (recall that we assume that the robot's pose is given) and stores the measurement vector in $\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}$ and the associated noise covariance matrix in $\Sigma_t^{\mathrm{r}_i,\mathbf{o}}$.

Line 7 computes $\mathcal{C}_t^{\mathrm{r}_i,\mathbf{o}}$ and $\mathcal{C}_t^{\mathrm{r}_i}$. Here, $\mathcal{C}_t^{\mathrm{r}_i,\mathbf{o}}$ represents the robot r_i's confidence on its observation measurement which can be calculated in various ways depending on the implementation and the scenario. In our implementation, we calculate this as :

$$\mathcal{C}_t^{\mathrm{r}_i,\mathbf{o}} \propto |\mathbf{\Sigma}_t^{\mathrm{r}_i,\mathbf{o}}|^{-1}, \tag{3.4}$$

Note that the observation measurement's noise covariance matrix $\mathbf{\Sigma}_t^{\mathrm{r}_i,\mathbf{o}}$ depends on several other factors based on the implementation's underlying object detection mechanism, e.g, distance and bearing to the object from the observing robot and the part of the object visible due to occlusions, facilitating the observation confidence measure to incorporate all those factors.

$\mathcal{C}_t^{\mathrm{r}_i}$ represents the robot r_i's confidence on its own pose obtained from its own self-localization mechanism, implemented separately from Algorithm 3.1. If, for example, the Monte Carlo localization (MCL) mechanism is used for the robot's self-localization, a good approach to do this is to consider the *number of effective particles* (see equation (4.1)) in the MCL's particle set as a measure of the robot's self-localization confidence[2] [44][45].

In line 12, the robot r_i receives $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$, $\Sigma_t^{\mathrm{r}_n,\mathbf{o}}$, $\mathcal{C}_t^{\mathrm{r}_n,\mathbf{o}}$ and $\mathcal{C}_t^{\mathrm{r}_n}$ from every other robot r_n in the team where $n = [1, ..., N]$ and $n \neq i$ (since r_i is the receiver robot itself). It is important to note that $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$ and $\Sigma_t^{\mathrm{r}_n,\mathbf{o}}$, obtained from any of the teammate robots, are already expressed in the world frame. This leads us to form a set $\mathbf{P}_t^{\mathrm{r}_i}$ which we call an observation measurement pool (OMP) for the robot r_i in the world frame. The elements of $\mathbf{P}_t^{\mathrm{r}_i}$ denote each team robot's individual observation measurements.

$$\mathbf{P}_t^{\mathrm{r}_i} = \{\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}} \mid n = 1, ..., N\} \tag{3.5}$$

In lines 10 and 13 of the Algorithm 3.1, we associate an element-weight (EW) to each

[2]In Chapter 4, where we present a novel cooperative robot localization technique, the self-localization confidence is discussed in more detail.

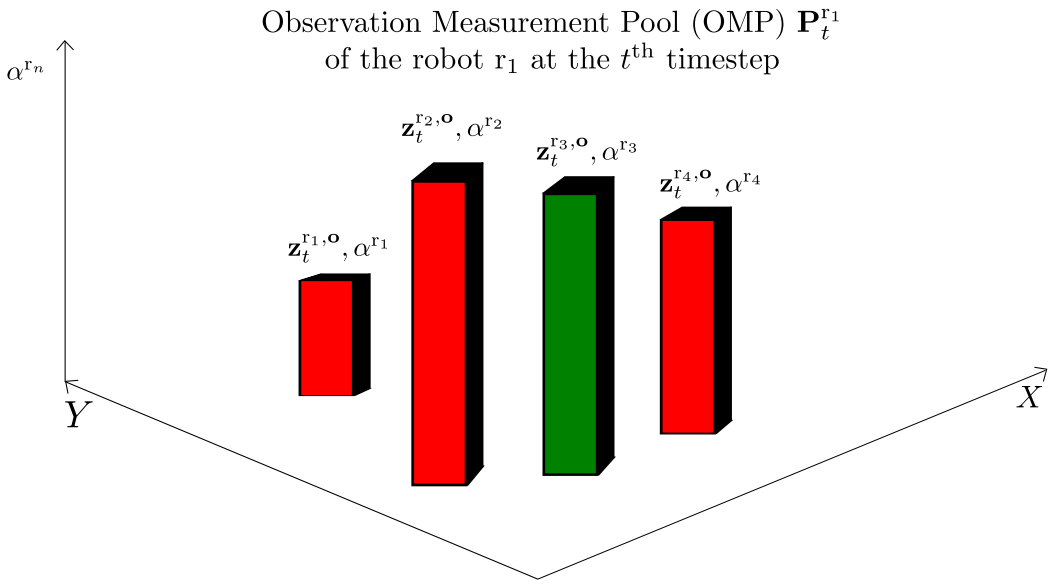


Figure 3.1: In the figure above, illustration of an OMP is presented for the following example. Consider a team of 4 robots cooperatively tracking an object in 2D space, each running its own instance of the Algorithm 3.1 and their object observation measurements are 2D vectors plus a Gaussian noise with zero mean and 2×2 covariance matrix. The figure above illustrates an OMP generated by the robot r_1 at the t^{th} timestep. The X and Y-axis represent the 2D world coordinates of the space in which the object is being tracked. The Z-axis represents α^{r_n}, calculated as per (3.6) and (3.7). In the figure above, the robot r_1 receives observation measurements from its teammate robots r_2, r_3 and r_4 and generates the OMP. The OMP elements $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$ are represented above as bar graphs positioned at the measurement vector represented by $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$ and height equal to their normalized element weights (EW) α^{r_n} for all $n = [1:4]$. Consider an m^{th} temporary particle $\bar{\mathbf{x}}_t^{[m],\mathbf{o}}$. For this particle, the OMP element $\mathbf{z}_t^{\mathrm{r}_3,\mathbf{o}}$ is drawn with probability α^{r_3} (hence the bar representing $\mathbf{z}_t^{\mathrm{r}_3,\mathbf{o}}$ is colored differently). Subsequently, the temporary particle's weight $\bar{w}_t^{[m],\mathbf{o}}$ is obtained as $\bar{w}_t^{[m],\mathbf{o}} \propto p(\mathbf{z}_t^{\mathrm{r}_3,\mathbf{o}} | \bar{\mathbf{x}}_t^{[m],\mathbf{o}})$. Similarly, for every other temporary particle, an OMP element will be drawn with probability equal to the element's EW and subsequently used to generate the temporary particle's weight.

element in the set $\mathbf{P}_t^{\mathrm{r}_i}$ (Note that the OMP element's EW, denoted further by the variable α^{r_n}, should not be confused with the m^{th} particle's weight denoted by $w_t^{[m],\mathbf{o}}$ elsewhere). For the robot r_i's OMP $\mathbf{P}_t^{\mathrm{r}_i}$, only the elements in it due to other teammates' observation measurements are weighted using their self-localization confidences and their observation

measurement confidences. The element in $\mathbf{P}_t^{\mathrm{r}_i}$ due to r_i's own observation measurement is weighted only by its observation measurement confidence. The reason for this is discussed later.

$$\alpha^{\mathrm{r}_i} = \mathcal{C}_t^{\mathrm{r}_i,\mathbf{o}} \tag{3.6}$$

$$\alpha^{\mathrm{r}_n} = \mathcal{C}_t^{\mathrm{r}_n,\mathbf{o}}\, \mathcal{C}_t^{\mathrm{r}_n}; \quad n = 1, ..., N;\ n \neq i \tag{3.7}$$

Lines 16–18 of the Algorithm 3.1 perform the OMP elements' EW normalization.

$$\alpha^{\mathrm{r}_n} = \frac{\alpha^{\mathrm{r}_n}}{\sum_{n=1}^{N} \alpha^{\mathrm{r}_n}}; \quad n = 1, ..., N \tag{3.8}$$

The most crucial step of the fusion is in lines 19–23 of the Algorithm 3.1. For every m^{th} temporary particle resulting from the prediction step of the algorithm, the following two steps are performed:

Step 1: Draw an element $\mathbf{z}_t^{\mathrm{r}_k,\mathbf{o}}$ from the OMP $\mathbf{P}_t^{\mathrm{r}_i}$ with probability α^{r_k}. Recall that $\mathbf{z}_t^{\mathrm{r}_k,\mathbf{o}}$ is the object observation measurement vector obtained by the robot r_k in the team where $k \in [1 : N]$.

Step 2: Calculate the weight $\bar{w}_t^{[m],\mathbf{o}}$ of the m^{th} temporary particle using the observation measurement $\mathbf{z}_t^{\mathrm{r}_k,\mathbf{o}}$ as $\bar{w}_t^{[m],\mathbf{o}} \propto p(\mathbf{z}_t^{\mathrm{r}_k,\mathbf{o}} | \bar{\mathbf{x}}_t^{[m],\mathbf{o}})$.

These two steps are further clarified in Figure 3.1 with the help of an illustration of the OMP.

Since the above two steps are performed for every temporary particle at a given iteration of the Algorithm 3.1, it fuses the information from all the elements of the OMP according to their respective EW. After this, in line 25, the particle set $\mathcal{X}_t^{\mathbf{o}}$ is obtained and returned as the output of the Algorithm 3.1 by resampling the temporary particle set $\bar{\mathcal{X}}_t^{\mathbf{o}}$. This can be done by any established method for resampling in the literature (see, e.g, low variance sampler in [43]). The association of the OMP elements to the temporary particles is done in every iteration of the PF-based cooperative tracker. Due to the sampling process in step 1, as mentioned above, a few temporary particles always get associated with the low EW elements of the OMP, which is a good way of maintaining the spread of the particles. This avoids the PF-based cooperative tracker from falling into the 'particle depletion' problem, which often occurs after resampling, where the general tendency of the particles is to get cloned to the highest weight temporary particle.

The PF-based cooperative tracker tackles the following issues in a generalized manner:

- The first problem which the tracker solves is that of a partial or complete occlusion. A partially occluded object's observation measurement by the robot r_i may lead to a low EW of the OMP element $\mathbf{z}_t^{\mathrm{r}_i,\mathbf{o}}$, generated by its own observation measurement. In case if the same object is being fully observed by another teammate robot r_n, it will lead to a high EW in the OMP element $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$, contributed by the robot r_n to the OMP $\mathbf{P}_t^{\mathrm{r}_i}$ of the robot r_i, and hence a greater chunk of particles will get associated to the OMP element $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$. Therefore, the robot r_i would still be able to make a good approximation of the target's posterior distribution after the resampling step.

- A problem that often spoils multi-robot cooperative perception is the poor quality of self-localization of the team robots. The OMP elements refer to the world frame. If a robot is wrongly localized, its observation of the object in the world frame will be incoherent with another correctly localized robot's observation of the same object in the world frame, although both might be observing the object correctly in their respective local frames. The incoherency here is due to the frame transformation carried out by the wrongly localized robot, of its observation in local frame to the world frame. In our approach this problem is solved by weighting the OMP elements by the associated robot's self localization confidence multiplied by its object observation measurement confidence. Also, this is done differently for the OMP elements corresponding to the recipient robot and the sender robots (3.6),(3.7). By doing this we ensure two major advantages:

  - A wrongly localized robot's good object observation measurement hardly influences other robot's OMP.
  - The wrongly localized robot would still have a high confidence in its own good observation, leading to a high EW of the OMP element due to its own observation measurement, which is necessary if we want to carry out visual tracking in its local frame. In such a task, it will not be affected by the object observation measurements of well localized teammates which are incoherent with the wrongly localized robot's local frame. This enables the wrongly localized robot to keep tracking the object with its local information during visual tracking, without relying on the incoherent world frame information.

- The third major advantage that we get in this tracker is that we do not have to deal with robot's motion directly. This is taken care *in situ* when we construct the

OMP in world frame, using the self-posture of the robot which inherently involves odometry or motion update of the robot.

It must be noted that the PF-based cooperative tracker does not, in any way, affect the self-localization estimates of the observing robots. The cooperative tracker estimates only the tracked object's position in the world frame for each robot that runs the tracker. It does not estimate the relative localization of the teammates. Each observing robot's self-localization needs to be obtained from a separate algorithm, e.g., Monte Carlo localization.

3.4 Implementation and Results

3.4.1 Testbed

We applied the algorithm proposed here to the robot soccer scenario. Our test-bed is the RoboCup Middle Sized League (MSL) (see Appendix A for a detailed description of the testbed). In robot soccer, one of the basic necessities is to continuously track the ball. A major concern is that the robots tend to lose their localization on the soccer field because of low-range vision and field symmetry. Moreover, since the field is too large as compared to a robot's camera vision field, a ball far from the robot ($> 4m$) is scarcely visible, and very often a nearby lying ball is occluded by a teammate or an opponent robot. Thus it becomes a very interesting and appropriate test-bed for our proposed algorithm. In order to evaluate the results of our approach's implementation we compare the results with the ground truth system (GTS) as explained in Appendix B.

From the datasets collected on the testbed (see Appendix A for the details of the datasets), we used the measurement logs of the dataset named '4_robots_dataset'. The measurement log of this dataset contains the odometry, 10 static landmark observation and the orange ball's observation measurements for the 4 robots: OMNI1, OMNI2, OMNI3 and OMNI4. From this log, along with the odometry and ball observation measurements, only 6 landmarks' observation measurements were used in the experiment described further in this section. Cooperative tracking results of the orange ball from all the four robots are presented.

3.4.2 Implementation

In our implementation, the individual robot's observation measurement is a 3D vector corresponding to the detected position (through the 3D detection mechanism presented in Chapter 2) of the ball in the global frame. The noise of this measurement is modeled as a

trivariate Gaussian with zero mean and covariance matrix $\Sigma_t^{\mathrm{r}_i,\mathbf{o}}$, which is a function of the observed distance to the ball and the part of the ball visible in the camera image.

To predict the ball's (target object) motion (lines 2–4 of the Algorithm 3.1), we use the approach (3.9) as in [20],

$$\bar{\mathbf{x}}_t^{[m],\mathbf{o}} = \mathbf{x}_{t-1}^{[m],\mathbf{o}} + \mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}}\Delta t + \mathbf{a}_t^{\mathrm{r}_i,\mathbf{o}}(\frac{\Delta t^2}{2}), \tag{3.9}$$

which is a constant velocity model with normally distributed acceleration noise about zero mean. $\bar{\mathbf{x}}_t^{[m],\mathbf{o}}$, as mentioned previously, is the m^{th} temporary particle's state vector component (recall that the other component of the m^{th} particle is its weight $\bar{w}_t^{[m],\mathbf{o}}$), representing the 3×1 state vector of the temporary particle's 3D position $\bar{\mathbf{x}}_t^{[m],\mathbf{o}} = [\bar{x}_t^{[m],\mathbf{o}}\ \bar{y}_t^{[m],\mathbf{o}}\ \bar{z}_t^{[m],\mathbf{o}}]^\top$. Δt is the time interval between the two consecutive timesteps $t-1$ and t (corresponding to the two consecutive iterations of the Algorithm 3.1). $\mathbf{v}_t^{\mathrm{r}_i,\mathbf{o}} = [v_{x_t}^{\mathrm{r}_i,\mathbf{o}}\ v_{y_t}^{\mathrm{r}_i,\mathbf{o}}\ v_{z_t}^{\mathrm{r}_i,\mathbf{o}}]^\top$ is the 3×1 velocity vector of the target object at the t^{th} timestep which is calculated using a separate velocity sensor based on a linear regression method over a number of previously estimated ball's positions. $\mathbf{a}_t^{\mathrm{r}_i,\mathbf{o}}$ is a 3×1 white zero mean random vector corresponding to an acceleration disturbance. For resampling we applied the low variance sampling method [43].

Since the PF-based cooperative tracker (Algorithm 3.1) is dependent on the self–localization confidence of the individual robots, the robot localization method implemented here is worth mentioning. Each robot uses the regular MCL algorithm [43] to estimate its own pose. To avoid notational and symbol complexity we skip the MCL algorithm's description in this chapter. However, since Chapter 4 deals with a modified MCL method for cooperative localization of robots, the regular MCL algorithm is presented there in the Algorithm 4.1 and the mechanism to obtain the robot localization confidence is given by (4.1) in that chapter.

Summarizing the implementation, we applied our proposed PF-based cooperative tracker (Algorithm 3.1) and the MCL-based localization (Algorithm 4.1), for each robot on the time-synchronized measurement logs. The logs of the ball observation measurements in 3D were obtained by applying Algorithm 2.1 on the raw images of the dataset '4_robots_dataset'.

3.4.3 Experiments and Results

In order to compare our cooperative tracker's performance we implemented a non-cooperative PF-based tracker on the same measurement logs (of the same dataset '4_robots_dataset'). The algorithm for the non-cooperative tracker on any robot resembles Algorithm 3.1 with

the fusion step replaced by a normal update step where the robot uses only its own observation measurement to weigh the particles.

In the video[3] accompanying this chapter we present the experiment's footage composed by concatenating the stream of images from one of the GTS cameras. Each robot has a different colored hat placed on top of it (OMNI1:Magenta, OMNI2:Brown, OMNI3:Red and OMNI4:Blue). The GTS uses these hats for detecting the robots' positions. The GT estimates by the GTS are marked in the video frames with an overlaid black circle around the robots' hat and the orange ball, except for the instances during which any of the robot's marker or the ball gets occluded from the GTS and it could not detect that robot's or the ball's position. The MCL estimates of the robot positions are marked in the video frames by overlaying a circle of the same color as the robot's hat. These circles are centred at the MCL's position estimate for the particular robot and are placed at the same height as that of the robot's hat from the ground level to facilitate easy visual comparison. The robot's orientation estimate by the MCL is marked by a radial line from the circle's center. Additionally, each robot's name along with a tag 'OK' or 'LOST' is continuously displayed above the robot's MCL estimated position. The tag is 'LOST' when its localization confidence falls below 0.5 and 'OK' when above 0.5. In this video we display the result of the Algorithm 3.1's implementation on OMNI1. The orange ball's 3D position estimates by the Algorithm 3.1 running on OMNI1 are marked by an orange circle overlaid on the video frames.

In Figure 3.2, we present a set of error plots comparing the results of the proposed cooperative approach with the comparative non-cooperative approach. The error is computed as the Euclidean distance between a tracker's estimated 3D position and the corresponding GTS's 3D estimate of the orange ball. From Figure 3.2, it can be visualized that when using the non-cooperative approach, none of the robots could track the ball for the total duration of the experiment, owing mainly to occlusions or because of being too far from the ball. Using our proposed cooperative approach this issue was not only solved but we also achieved an overall higher tracking accuracy which can be seen in the Table 3.2, where the percentage reduction of the mean tracking error by all the robots is presented. A notable exception is in the case of OMNI1, where we see a slight increase in the world frame mean error when using cooperative tracking over the non-cooperative approach (notice that the reduction of the world frame mean error for OMNI1 is -8.46% in the Table 3.2), but a substantial decrease in its local frame mean error of the tracked ball's position (The re-

[3]Video of the experiment: Application of the Algorithm 3.1 on '4_robots_dataset'.
http://users.isr.ist.utl.pt/∼aahmad/PhDThesisVideos/Chap_3_MCOT/PF_MCOT.mpg

duction of the local frame mean error for OMNI1 is 36.86% in the Table 3.2). The reason is as follows. OMNI1's own observation measurements of the ball are better than all other robots as well as the ball is within its vision range for $\sim 90\%$ of the experiment's duration. When performing the cooperative tracking, OMNI1's benefit lies in tracking the ball during the rest $\sim 10\%$ of the experiment duration (during which the ball was out of its vision field but in the vision field of any teammate robot). However, since the teammates' observation measurements are received by OMNI1 for the whole duration of the experiment during the cooperative tracking, a slight importance is given to the OMP elements due to them. This leads OMNI1 to incorporate more noisy observation measurements in addition to its own less noisy observation measurements causing a slight overall increase in the world frame error of its tracked ball position. Nevertheless, since the teammates of OMNI1 receive the less noisy observations from it, they are able to substantially reduce the world frame errors of their tracked ball positions (supported by the fact that all other teammates' reduction of mean tracking errors are positive in the Table 3.2).

Table 3.1 presents the statistics of all the errors displayed in the plots of the Figure 3.2. It must be noted that the variance of error achieved by the non-cooperative tracker, apart from being low, is further reduced by ~ 15 to 40% when using the proposed cooperative approach, denoting a more consistent tracking throughout the experiment.

The errors in the plots of Figure 3.2 are computed for each robot's tracker's world frame estimates of the ball position, hence these errors are offset by the tracking robot's localization errors. Since the GTS, apart from estimating the ball's world position, also estimates the robots' positions (but not their orientations), it was possible to compute the range error of the ball's tracked position in each tracking robot's local frame. A histogram representation of the local frame range errors for each robot's cooperative and the non-cooperative PF-based tracker is presented in the Figure 3.3. The statistics of these errors are presented in the Table 3.1 alongside the statistics of the world-frame errors. Note that both the cooperative and the non-cooperative PF-based tracker were implemented on the exact same datasets/measurement logs. The median of the local frame range errors gets significantly reduced when using the cooperative approach over the non-cooperative tracking and is consistent with the histogram plots in the Figure 3.3 for the same.

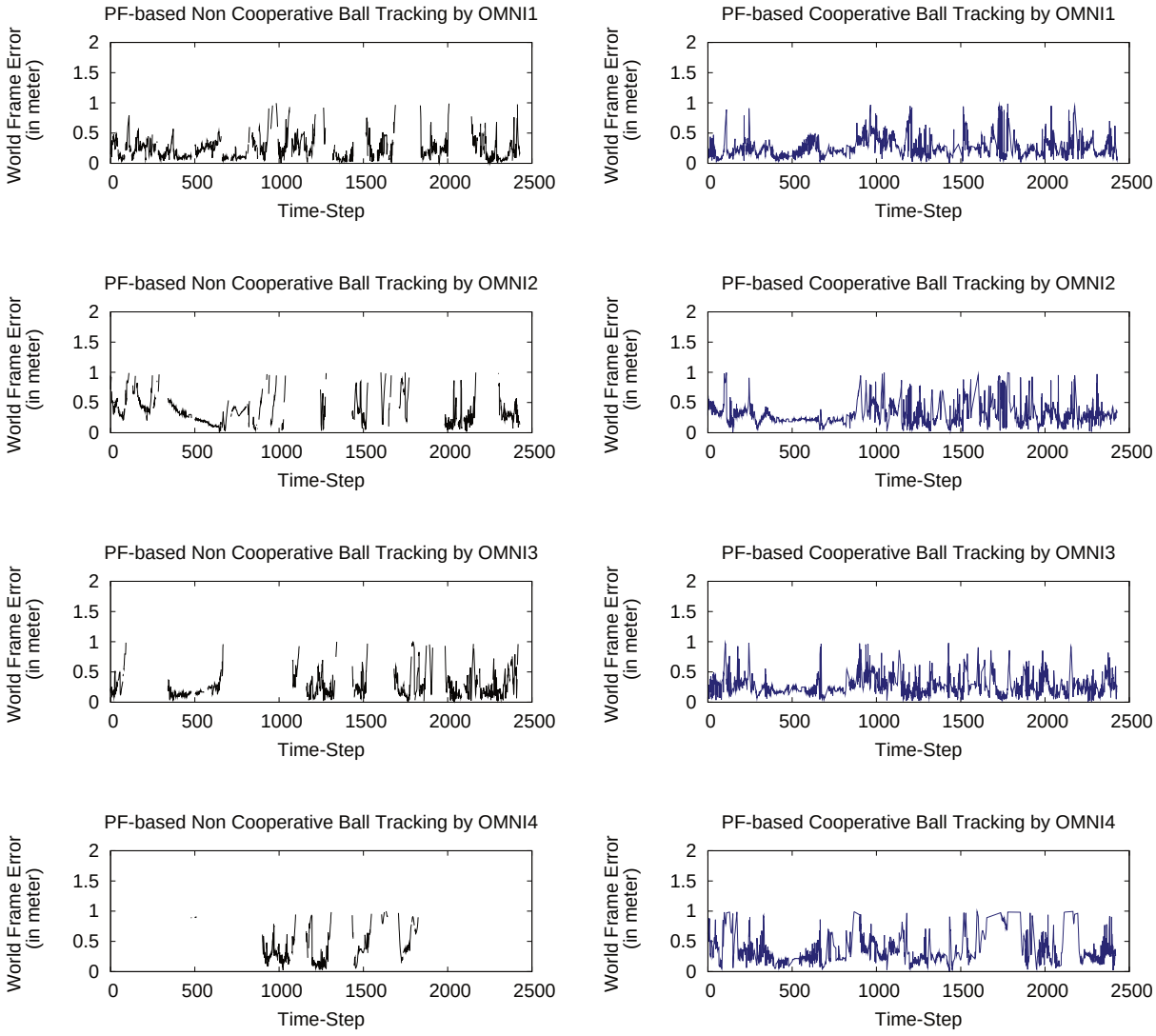


Figure 3.2: The plots in the left column present the orange ball world-frame tracking error by each robot's PF-based non-cooperative tracker in 3D. The corresponding plots on the right column present the same for the PF-based cooperative tracker (Algorithm 3.1). Both tracker's object observation measurements were done using the Algorithm 2.1. If any tracker loses the ball, the error is not computed and its plot omitted.

	Non-Cooperative						**Cooperative**					
	World Frame Position Error			**Local Frame Range Error**			**World Frame Position Errors**			**Local Frame Range Error**		
Robot	Mean (m)	Median (m)	Var (m^2)	Mean (m)	Median (m)	Var (m^2)	Mean (m)	Median (m)	Var (m^2)	Mean (m)	Median (m)	Var (m^2)
OMNI1	0.248	0.209	0.033	0.255	0.172	0.068	0.269	0.226	0.029	0.161	0.089	0.037
OMNI2	0.368	0.325	0.048	0.317	0.252	0.078	0.316	0.263	0.034	0.237	0.152	0.055
OMNI3	0.280	0.207	0.045	0.272	0.186	0.058	0.275	0.233	0.027	0.200	0.147	0.033
OMNI4	0.424	0.354	0.070	0.281	0.168	0.065	0.353	0.282	0.047	0.266	0.202	0.055

Table 3.1: Error statistics of the tracked ball's 3D position estimates by each robot.

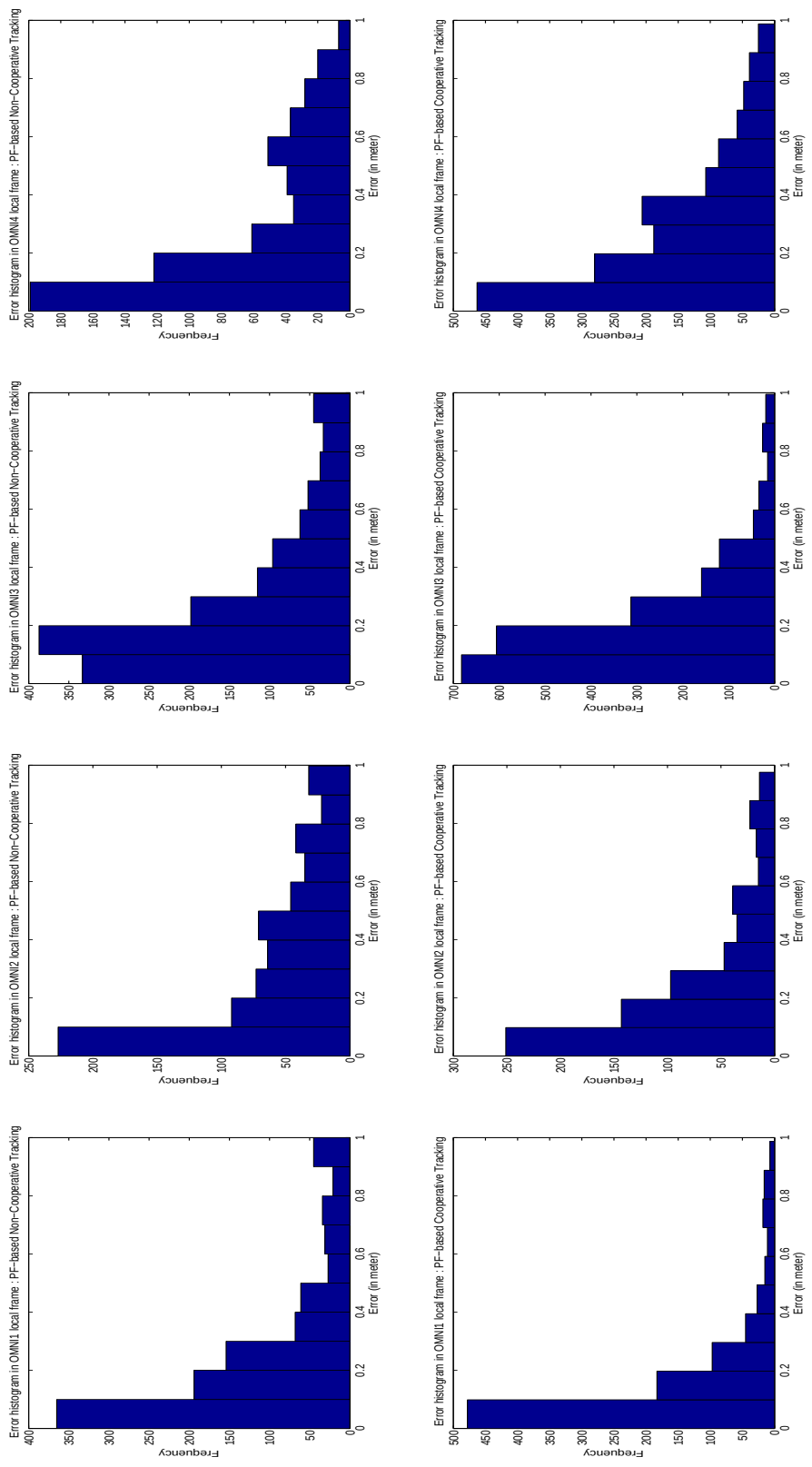


Figure 3.3: The plots in the top row present the histograms of the tracked ball's range errors in each robot's local frame when using a PF-based non-cooperative tracker. The plots in the bottom row display the histograms of the tracked ball's range errors in each robot's local frame when using the proposed PF-based cooperative tracker. In both cases, the same dataset and its measurement logs were used.

	Reduction of Position Error (%)	
Robot	World Frame	Local Frame
OMNI1	-8.46	36.86
OMNI2	14.13	25.23
OMNI3	1.78	26.47
OMNI4	16.74	5.33

Table 3.2: Percentage reduction of mean error when using the proposed cooperative approach over the non-cooperative approach for tracking.

During the experiment, except for OMNI4, all the robots were well localized $\sim 95\%$ of the time. This can be visualized in the accompanying video where the robots' localization estimates are marked. OMNI4, owing mainly to its spurious landmark observations and a high odometry noise, often remained poorly localized. Furthermore, as it can be seen in the experiment's footage, the ball was often away from OMNI4. As a consequence, when running the non-cooperative tracker, OMNI4 could not track the ball for most of the time but was able to do so when running the cooperative tracker, which is also supported by the error plots for OMNI4 in Figure 3.2. It must be noted that for those few instances when OMNI4 had the ball in its own vision field, while being poorly localized, its own ball observation measurements did not significantly affect the other robots' cooperative tracking.

Because of its poor localization and not observing the ball itself most of the time, OMNI4 could only slightly benefit in its local frame tracking by using our proposed cooperative approach. This is supported by the fact that the local frame range error reduction was only $\sim 5\%$ compared to the non-cooperative tracking approach (Table 3.2). Nevertheless, it could track the ball well in the world frame. This is supported by the fact that the ball observation measurements it received was from the other well-localized robots, facilitating OMNI4 to reduce the mean of its world frame tracking error by $\sim 16\%$.

3.5 Summary

In this chapter, we introduced a particle filter-based algorithm for cooperative object tracking by a team of mobile robots. We implemented the algorithm for cooperatively tracking a spherical object (soccer ball) in 3D space by a team of soccer robots. Supported by the real robots experiments and their results, we conclude that our algorithm provides a robust approach for tracking an object cooperatively by a team of mobile robots. A few major

points can be enumerated:

- The approach is a solution for continuously tracking an object which is likely to be occluded or partially occluded quite often.
- If the object is confidently located by a wrongly localized robot, after fusion it would track it correctly in its own local frame and affect other teammates' fused observation measurements quite insignificantly.
- By sharing a compact representation of the observation measurements (in which only the measurements' parameters are enough to construct its compact representation), we significantly reduce the use of bandwidth and communication time which facilitates real-time tracking of the object.
- Reference frame-incoherent object observation measurements across the teammates are taken into account within a common framework.
- Self-localization confidence of the observing mobile robots are taken into account.

Ongoing work includes extending our method of cooperative tracking to multiple sensors on the same robot as well as to relax colored object tracking assumptions. Tracking a random colored object in 3-D space will require a different observation measurement model without any changes in our current cooperative tracker. This is because the Algorithm 3.1 is independent of measurement models used.

3.6 Related Publications

The work presented in this chapter was initially published [9] in the European Conference on Mobile Robotics (ECMR) 2011, Örebro, Sweden (submitted: April 2011, accepted: June, 2011). It was later voted as one of the best papers presented in the conference and invited for a special issue article in the Robotics and Autonomous Systems (RAS) journal, where it was accepted for publication [10].

Chapter 4

Multi-Robot Cooperative Robot Localization

4.1 Introduction

In this chapter, we introduce a modification of Monte-Carlo Localization (MCL) algorithm that changes the particle spreading step (used when a robot detects it is lost), using information provided by other robot(s) of the team on the location of an object commonly observed by the lost robot and the other robot(s). This modification speeds up the recovery of the lost robot and is robust to perceptual aliases, namely when environments have symmetries, due to the extra information provided by the teammates. The introduced method enables cooperative localization in a multirobot team, using visually shared objects, taking advantage of the specific features of particle filter (PF) algorithms. Each robot is assumed to run MCL for its self-localization, and to be able to detect when the uncertainty about its localization drops below some threshold. An observation model that enables determining the level of confidence on the tracked object position estimate is also assumed to be available at each robot of the team. Though these assumptions are, to some extent, stronger than those assumed by cooperative simultaneous localization and mapping methods, they allow global robot and object localization. Though other authors have explored the use of observations to initialize and/or reset particle filters adequately [46],[47], the use of shared observations of common objects to cooperatively improve each team robot's localization using MCL is novel, to the best of our knowledge.

The chapter is organized as follows: in Section 4.2, we describe our cooperative localization method. Results of experiments with real soccer robots in the RoboCup Middle-Size League (MSL), that use a soccer ball as the visually shared object, are presented in Sec-

tion 4.3. A summary of the chapter is presented in Section 4.4.

4.2 Cooperative Localization Using a Visually Shared Object

Let us consider a team of N robots, $\mathrm{r}_1, \ldots, \mathrm{r}_N$. Robot r_i has pose (position + orientation) coordinates $\mathcal{L}_t^{\mathrm{r}_i} = [x_t^{\mathrm{r}_i} \;\; y_t^{\mathrm{r}_i} \;\; \theta_t^{\mathrm{r}_i}]^\top$ in a global world frame at the t^{th} timestep, and estimates them using an MCL algorithm.

Each robot can determine the position of an object $\mathbf{O}$ in its local frame, therefore being able to determine its distance and bearing to that object as well. Robots can also determine if they are lost or kidnapped, i.e., if their confidence in the pose estimate drops below some threshold. If a robot is not lost, it can also determine the object position in the global world frame using the transformation between its local frame and the global world frame that results from the knowledge of its pose. The estimate of the object position in any frame is determined based on a probabilistic measurement model that includes the uncertainty about the actual object position. When the global world frame is used, additional uncertainty is caused by the uncertain pose of the observing robot.

The object is assumed to lie always on the ground plane where the robot moves. The position of the object as determined by robot r_i at the t^{th} timestep in the global world frame is denoted by $\mathbf{O}_t^{\mathrm{r}_i} = [x_t^{\mathbf{o},\mathrm{r}_i} \;\; y_t^{\mathbf{o},\mathrm{r}_i}]^\top$, while the distance and bearing of the object with respect to the robot, as measured by the robot, are given by $d_t^{\mathbf{o},\mathrm{r}_i}$ and $\psi_t^{\mathbf{o},\mathrm{r}_i}$, respectively.

4.2.1 Overall Description

The original MCL algorithm [43] used by a robot r_i in the team to estimate its pose is presented in Algorithm 4.1.

$\mathcal{X}_{t-1}^{\mathrm{r}_i} \triangleq \{\langle \mathbf{x}_{t-1}^{[m],\mathrm{r}_i}, w_{t-1}^{[m],\mathrm{r}_i} \rangle\}_{m=1}^{M}$ is the set of M particles resulting from the immediate previous timestep $t-1$ of the algorithm's iteration process. $\mathbf{u}_t^{\mathrm{r}_i}$ is the robot r_i's odometry reading at timestep t. $\mathbf{z}_t^{\mathrm{r}_i}$ is the robot's observation measurements at timestep t, concerning its self-localization. *map* is a map of the robot world (e.g., a set of landmarks or other) and w_{fast}, w_{slow} are auxiliary particle weight averages, with $0 \leqslant \alpha_{slow} \ll \alpha_{fast}$, such that w_{slow} provides long-term averages and w_{fast} provides short-term averages. The algorithm uses a ***sample_motion_model*** and a ***measurement_model*** to update, at each timestep, the robot pose, from the odometry $\mathbf{u}_t^{\mathrm{r}_i}$ and observation measurements $\mathbf{z}_t^{\mathrm{r}_i}$ information and stores it in a temporary particle set $\bar{\mathcal{X}}_t^{\mathrm{r}_i} \triangleq \{\langle \bar{\mathbf{x}}_t^{[m],\mathrm{r}_i}, \bar{w}_t^{[m],\mathrm{r}_i} \rangle\}_{m=1}^{M}$. It adds random particles to $\mathcal{X}_t^{\mathrm{r}_i}$,

Algorithm 4.1 : MCL$(\mathcal{X}_{t-1}^{\mathrm{r}_i}, \mathbf{u}_t^{\mathrm{r}_i}, \mathbf{z}_t^{\mathrm{r}_i}, map)$

static w_{slow}, w_{fast}
$\bar{\mathcal{X}}_t^{\mathrm{r}_i} = \mathcal{X}_t^{\mathrm{r}_i} = \mathrm{NULL}$
$w_{avg} = 0$
for $m = 1$ to M **do**
  $\bar{\mathbf{x}}_t^{[m],\mathrm{r}_i} = \boldsymbol{sample\_motion\_model}\ (\mathbf{x}_{t-1}^{[m],\mathrm{r}_i}, \mathbf{u}_t^{\mathrm{r}_i})$
  $\bar{w}_t^{[m],\mathrm{r}_i} = \boldsymbol{measurement\_model}(\bar{\mathbf{x}}_t^{[m],\mathrm{r}_i}, \mathbf{z}_t^{\mathrm{r}_i}, map)$
  $\bar{\mathcal{X}}_t^{\mathrm{r}_i} = \bar{\mathcal{X}}_t^{\mathrm{r}_i} + \langle \bar{\mathbf{x}}_t^{[m],\mathrm{r}_i}, \bar{w}_t^{[m],\mathrm{r}_i} \rangle$
  $w_{avg} = w_{avg} + \frac{1}{M} \bar{w}_t^{[m],\mathrm{r}_i}$
end for
$w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow})$
$w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast})$
for $m = 1$ to M **do**
  with probability $\max\{0.0, 1 - w_{fast}/w_{slow}\}$ **do**
  add random pose to $\mathcal{X}_t^{\mathrm{r}_i}$
  else
  draw $k \in \{1, ..., M\}$ with probability $\propto \bar{w}_t^{[k],\mathrm{r}_i}$
  add $\langle \bar{\mathbf{x}}_t^{[k],\mathrm{r}_i}, \bar{w}_t^{[k],\mathrm{r}_i} \rangle$ to $\mathcal{X}_t^{\mathrm{r}_i}$
  end with
end for
return $\mathcal{X}_t^{\mathrm{r}_i}$

the particle set obtained after applying re-sampling step on $\bar{\mathcal{X}}_t^{\mathrm{r}_i}$ (where the probability of cloning an existing temporary particle is proportional to its weight), with a probability that increases with the deviation of the short term w_{fast} average from the long-term w_{slow} average. In the limit case, when all the temporary particle weights tend to zero in the short-term, all particles are reset according to a uniform distribution.

When cooperative localization in a multirobot team, using visually shared objects, is intended, MCL running in a given robot r_i must be modified so as to use information from other robot(s) in the team, when w_{fast}/w_{slow} drops below a given confidence threshold $C_{threshold}$, meaning that r_i is lost or was kidnapped. That information comes in the form of the object position determined by the other robot(s). Assuming r_i (the lost/kidnapped robot) can observe the same object, the re-sampling is then based on a spatial probability distribution which depends on the distance and bearing of the lost robot to the object and on the uncertainty associated to the object position measurement provided by the

other robot(s). This way, while a uniform distribution is still used to keep a certain level of exploration of the pose space to make the algorithm robust to measurement and motion errors, if those errors influence becomes too high, the particles are completely reset according to the cooperative information from teammates about a visually shared object.

The new cooperative MCL algorithm used by robot r_i from the team to estimate its pose is presented in Algorithm 4.2, where the right superscript index r_i is used for local estimates, odometry readings, observations and the particle sets.

Algorithm 4.2 : Cooperative_Shared_Object_MCL$(\mathcal{X}_{t-1}^{\mathrm{r}_i}, \mathbf{u}_t^{\mathrm{r}_i}, \mathbf{z}_t^{\mathrm{r}_i}, map)$

static w_{slow}, w_{fast}
$\bar{\mathcal{X}}_t^{\mathrm{r}_i} = \mathcal{X}_t^{\mathrm{r}_i} = \mathrm{NULL}$
$w_{avg} = 0$
for $m = 1$ to M **do**
  $\bar{\mathbf{x}}_t^{[m],\mathrm{r}_i} = \boldsymbol{sample\_motion\_model}\ (\mathbf{x}_{t-1}^{[m],\mathrm{r}_i}, \mathbf{u}_t^{\mathrm{r}_i})$
  $\bar{w}_t^{[m],\mathrm{r}_i} = \boldsymbol{measurement\_model}(\bar{\mathbf{x}}_t^{[m],\mathrm{r}_i}, \mathbf{z}_t^{\mathrm{r}_i}, map)$
  $\bar{\mathcal{X}}_t^{\mathrm{r}_i} = \bar{\mathcal{X}}_t^{\mathrm{r}_i} + \langle \bar{\mathbf{x}}_t^{[m],\mathrm{r}_i}, \bar{w}_t^{[m],\mathrm{r}_i} \rangle$
  $w_{avg} = w_{avg} + \frac{1}{M} \bar{w}_t^{[m],\mathrm{r}_i}$
end for
$w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow})$
$w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast})$
if $w_{fast}/w_{slow} < C_{threshold}$ **and** info about object position in global world frame available from teammate(s) $\mathrm{r}_j \neq \mathrm{r}_i$ **and** object visible to r_i **then**
  draw $\mathcal{X}_t^{\mathrm{r}_i}$ according to object pose spatial probability distribution determined from r_i and r_j's information
else
  for $m = 1$ to M **do**
    with probability $\max\{0.0, 1 - w_{fast}/w_{slow}\}$ **do**
    add random pose to $\mathcal{X}_t^{\mathrm{r}_i}$
    else
    draw $k \in \{1, ..., M\}$ with probability $\propto \bar{w}_t^{[k],\mathrm{r}_i}$
    add $\langle \bar{\mathbf{x}}_t^{[k],\mathrm{r}_i}, \bar{w}_t^{[k],\mathrm{r}_i} \rangle$ to $\mathcal{X}_t^{\mathrm{r}_i}$
    end with
  end for
end if
return $\mathcal{X}_t^{\mathrm{r}_i}$

In the **Cooperative_Shared_Object_MCL** algorithm one needs to further detail how to handle the following issues:

1. Information about object position in global world frame available from teammate(s) $\mathrm{r}_j \neq \mathrm{r}_i$;
2. How to draw $\mathcal{X}_t^{\mathrm{r}_i}$ according to object pose spatial probability distribution determined from r_i and r_j's information.

The first issue concerns $\mathbf{O}_t^{\mathrm{r}_j} = [x_t^{\mathbf{o},\mathrm{r}_j} \;\; y_t^{\mathbf{o},\mathrm{r}_j}]^\top$, i.e., the object position in the global world frame, as determined by the robot r_j (in general, r_j may be any robot but r_i, or several such robots, in which case the object position from one of them is selected depending on whether that robot is well localized or not). Furthermore, we assume that, associated with $\mathbf{O}_t^{\mathrm{r}_j}$, r_j provides a confidence measure regarding that information. That confidence measure depends on r_j's

- object observation model and
- self-localization estimate uncertainty.

Assuming a bivariate Gaussian object observation model for r_j centered on $\mathbf{O}_t^{\mathrm{r}_j}$ and with a covariance matrix $\mathbf{\Sigma}_t^{\mathbf{o},\mathrm{r}_j}$ dependent on the distance and bearing to the object from r_j, this item contributes to the confidence measure with $|\mathbf{\Sigma}_t^{\mathbf{o},\mathrm{r}_j}|^{-1}$.

The self-localization estimate confidence factor of the robot r_j must be determined from the particle set of the **Cooperative_Shared_Object_MCL** algorithm running on the robot r_j. One good approach to do this, as presented by the authors in [44][45], is to consider the *number of effective particles* (denoted further by $\mathcal{N}_t^{\mathrm{r}_j}$) in r_j's particle set $\mathcal{X}_t^{\mathrm{r}_j}$. This is given by

$$\mathcal{N}_t^{\mathrm{r}_j} = \frac{1}{\sum_{m=1}^{M} (w_t^{[m],\mathrm{r}_j})^2} \tag{4.1}$$

after the weights $\{w_t^{[m],\mathrm{r}_j}\}_{m=1}^{M}$ are normalized.

Considering both factors, the measure of confidence $\mathcal{C}_t^{\mathrm{r}_j,\mathbf{o}}$ of the robot r_j on its own estimate of the object's position $\mathbf{O}_t^{\mathrm{r}_j}$ is given by

$$\mathcal{C}_t^{\mathrm{r}_j,\mathbf{o}} = \eta(\mathcal{N}_t^{\mathrm{r}_j} |\mathbf{\Sigma}_t^{\mathbf{o},\mathrm{r}_j}|^{-1}) \tag{4.2}$$

where η is a normalization factor.

Regarding the second issue and assuming that the object is visible to the lost robot r_i, this robot determines the distance and bearing of the object in its local frame, $(d_t^{\mathbf{o},\mathrm{r}_i}, \psi_t^{\mathbf{o},\mathrm{r}_i})$.

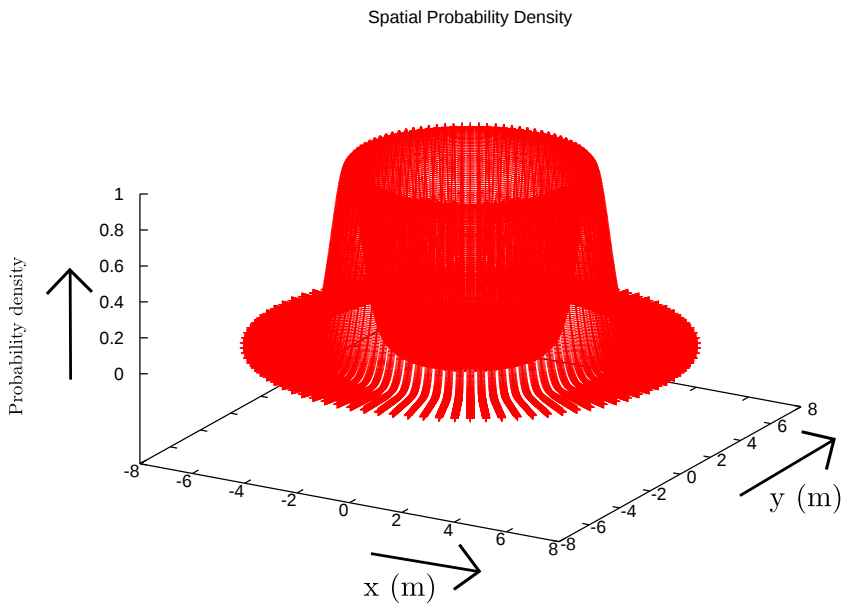


Figure 4.1: Typical spatial probability density function from which particles are drawn, after a decision to reset MCL.

The distance $d_t^{\mathbf{o},\mathrm{r}_i}$ is used to parametrize the spatial probability density function (pdf) from which particles representing the robot position in polar coordinates are drawn, after a decision to reset MCL. This bivariate (using polar coordinates d and ψ centered at $\mathbf{O}_t^{\mathrm{r}_j}$, the global world position of the object as estimated by the robot r_j and communicated to the robot r_i) pdf is:

- Gaussian in the d variable, with mean value $d_t^{\mathbf{o},\mathrm{r}_i}$ and variance inversely proportional to the confidence factor $\mathcal{C}_t^{\mathrm{r}_j,\mathbf{o}}$;
- uniform in the ψ variable, in the interval $[0, 2\pi[$ rad.

An example of this pdf is shown in Figure 4.1.

One can trivially map the polar coordinates onto Cartesian coordinates, thus obtaining $x_t^{[m],\mathrm{r}_i}$ and $y_t^{[m],\mathrm{r}_i}$, the position components of the m^{th} particle $\mathbf{x}_t^{[m],\mathrm{r}_i}$ in the resultant particle set $\mathcal{X}_t^{\mathrm{r}_i}$.

The orientation component $\theta_t^{[m],\mathrm{r}_i}$ of $\mathbf{x}_t^{[m],\mathrm{r}_i}$ is computed from the bearing angle $\psi_t^{\mathbf{o},\mathrm{r}_i}$ of the object, i.e., the angle between the robot r_i's longitudinal axis (the one pointing towards

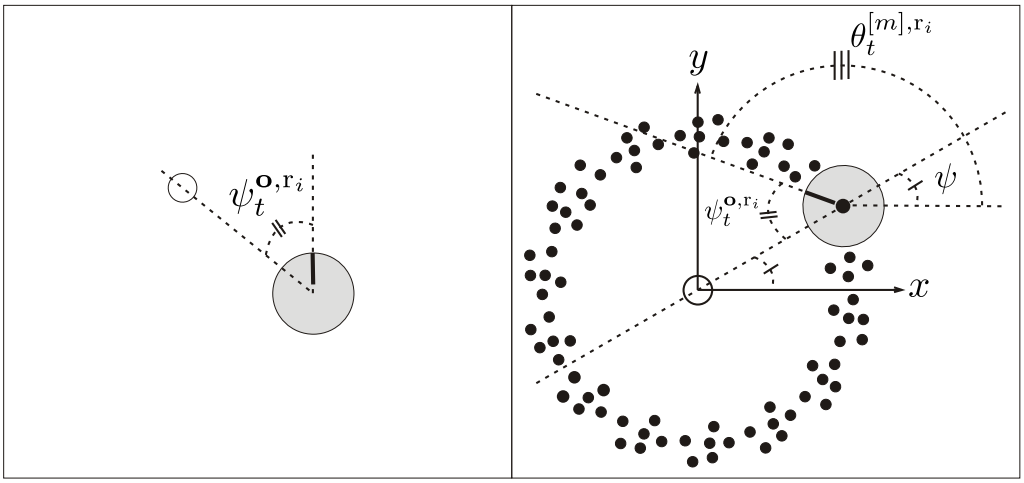


Figure 4.2: Computing the orientation of an m^{th} particle representing a robot pose hypothesis. On the left, bearing of the object with respect to the robot. On the right: relevant angles for the computation of the orientation component of the m^{th} particle as per (4.1).

its "front") and the line connecting its center with the object center (see Figure 4.2 - left), and the actual angle ψ of this line with respect to the x-axis of a frame centered on the object, i.e., the particle angle in polar coordinates centered on the object (see Figure 4.2 - right). We add random noise θ_{rand} with a zero mean Gaussian pdf representing the bearing measurement error model. Hence, from Figure 4.2:

$$\theta_t^{[m],\mathrm{r}_i} = \psi + \pi - \psi_t^{\mathbf{o},\mathrm{r}_i} + \theta_{rand}. \tag{4.3}$$

In summary, the **Cooperative_Shared_Object_MCL** algorithm, presented in Algorithm 4.2, modifies the plain MCL Algorithm 4.1, replacing the "standard" particle reset, using a spatially uniform pdf, by a particle reset based on the information about the position, in the global world frame (determined by one or more teammates), and the distance and bearing, in the local frame of the lost/kidnapped robot, of an object visible to all the intervening robots. Additional input parameters of this algorithm are (assuming r_i as the lost/kidnapped robot and r_j as any of the robots providing information to improve its localization):

- $C_{threshold}$ to determine if the robot is lost or was kidnapped;
- determinant of the object measurement model covariance matrix $|\mathbf{\Sigma}_t^{\mathbf{o},\mathrm{r}_j}|$, sent by r_j to r_i when r_i detects it is lost and requests support from teammates;

- the *number of effective particles* $\mathcal{N}_t^{\mathrm{r}_j}$ in r_j's **Cooperative_Shared_Object_MCL** algorithm, sent by r_j to r_i when r_i detects it is lost and requests support from teammates;

- distance and bearing of the object in r_i's local frame, $(d_t^{\mathbf{o},\mathrm{r}_i}, \psi_t^{\mathbf{o},\mathrm{r}_i})$, measured by r_i when it is lost and receives the above information from teammate(s);

- variance of the zero mean Gaussian pdf representing the bearing measurement error model at r_i (see previous item).

The regular procedure for each of the team robots is to run Algorithm 4.2. When $w_{fast}/w_{slow} < C_{threshold}$ at r_i, this robot requests for help from the teammates. One or more teammates r_j send the object position in global world coordinates and the associated confidence factor $\mathcal{C}_t^{\mathrm{r}_j,\mathbf{o}}$. Then, r_i's particles are spread uniformly over a circle centered on the object, with nominal radius equal to $d_t^{\mathbf{o},\mathrm{r}_i}$, the distance to the object measured by r_i, added to a Gaussian uncertainty around this value, with variance proportional to $\mathcal{C}_t^{\mathrm{r}_j,\mathbf{o}}$. The orientation component of the particle results from the bearing $\psi_t^{\mathbf{o},\mathrm{r}_i}$ of the object measured by r_i in its local frame, including an uncertainty proportional to the variance of the Gaussian representing the bearing measurement model.

Practical issues to be considered are:

- After a robot detects it is lost and spreads its particles over a circle centered with the visually shared object, it should run the regular MCL algorithm (Algorithm 4.1) in the next steps (a number of steps dependent on the application), so that it does not keep resetting its particles over a circle around the object, while its pose estimate has not converged to the actual value;

- The decision on which teammate(s) can contribute with useful information can be taken considering their confidence factor(s) $\mathcal{C}_t^{\mathrm{r}_j,\mathbf{o}}$, and only using the information provided by the robot with the highest $\mathcal{C}_t^{\mathrm{r}_j,\mathbf{o}}$ (if above some given threshold). In general, all teammates can contribute, but in some cases their information may be highly uncertain.

4.2.2 Particle Spreading Validation

There is a specific situation where the proposed algorithm requires some improvement, e.g., when a robot is kidnapped to a pose where it observes the object at approximately the same distance of where the robot was before. In this case, when the robot detects

it is lost by checking its w_{fast}/w_{slow} value, it will still spread new particles over a circle centered with the object, including a region around where the robot wrongly estimates its pose. To prevent such situations, the algorithm must include a restriction that requires all the re-spread particles to be out of a region (e.g., a circle) that includes the majority of the particles at the MCL step when the robot detected it was lost. Nonetheless, it is important to note that particles may be correct in the old pose, if they just have similar positions but fairly different orientations. Because of this, the algorithm must also check if the orientation hypothesis associated to each particle is within a small range of values around the orientation at kidnapping detection time. If they are not, the restriction above does not apply.

4.3 Results of Implementation in Real Soccer Robots

We have applied the **Cooperative_Shared_Object_MCL** (Algorithm 4.2) to real robots in RoboCup Soccer Middle-Size League (MSL), in which the robots use the ball to regain their pose when they are kidnapped and detect to be lost on the field. Figure 4.3 provides an example that illustrates the algorithm application in this scenario.

4.3.1 Experimental Setup 1

In this setup, tests were made by kidnapping a robot to nine different positions, in one quarter of the whole soccer field, as depicted in Figure 4.4. The other field regions would not provide extra information, due to the soccer field symmetry. One teammate stopped in a random position always sees the ball and informs the kidnapped robot of the ball's position on the global field frame. Both robots use MCL with 1000 particles, as described in a previous paper [21]. The standard deviation of the Gaussian used to model the bearing angle measurement error at the kidnapped robot was adjusted experimentally as $\pi/12$ rad.

We kidnapped the robot five times for each of the nine positions. Kidnapping was carried out by picking up the robot and moving it to another location with MCL on and after its convergence to a correct estimate. After kidnapping, the increase of uniformly distributed particles was visible, turning quickly to a re-spread over circle centered with the ball position, as estimated by the teammate.

The algorithm runs on a NEC Versa FS900 laptop with a Centrino 1.6GHz processor and 512Mb of memory, using images provided by a Marlin AVT F033C firewire camera, and in parallel with the other processes used to make the robot play soccer. The camera

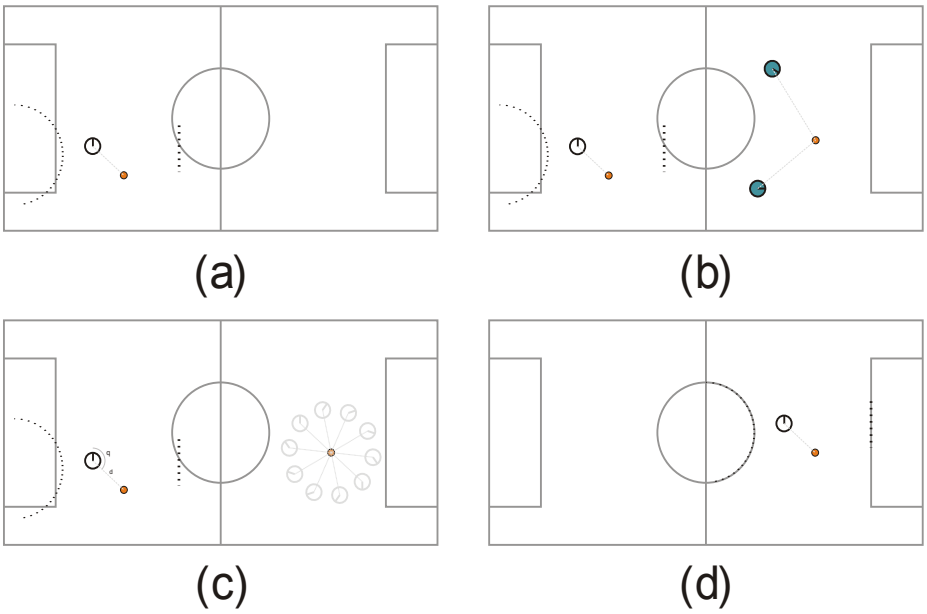


Figure 4.3: Cooperative robot localization in RoboCup Soccer MSL: (a) The white robot determines the ball position in its local frame but its estimated pose (based on field line detection - lines observed by robot are dashed and do not coincide at all with the actual solid field lines) is incorrect, because the robot was kidnapped. (b) Teammates (green robots) communicate the ball position in the global world frame, as well as the corresponding confidence factor. (c) Lost robot measures its distance and bearing to the ball and re-spreads the particles according to this and to the ball position team estimate. (d) The previously lost robot regains its correct pose.

is mounted as part of an omnidirectional dioptric system, using a fish-eye lens attached to it.

Figure 4.5 shows the actual position of the robots disposed as in position 4 of Figure 4.4.

4.3.2 Experimental Setup 2

In this setup, the teammate which observes the ball tracks and follows the ball by maintaining a fixed orientation and distance with respect to ball. We kidnapped the other robot in the following 4 cases during this setup:

- Case 1: Both observer robot and the ball moving when the ball is in the field of view (FOV) of both robots. Robot kidnapped twice in this case.

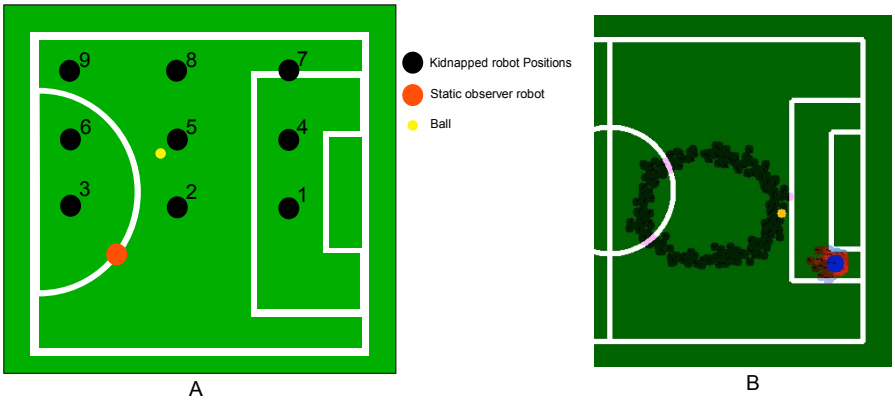


Figure 4.4: Layout of experiments. A) The black numbered spots correspond to the positions to where one of the team robots was kidnapped. The red circle represents a static robot, always well localized, and watching the ball (smaller yellow circle). B) The figure in B is an example snapshot of the global frame interface which plots real robot and ball positions as well as particles used by MCL in real-time. Here it shows the kidnapped robot spreading particles after getting lost and using the shared ball

Field position	Successes	Iterations to converge
1	5	14.6
2	5	14.8
3	5	7.2
4	5	19.5
5	4	25.5
6	4	10
7	4	15.3
8	5	21
9	5	16

Table 4.1: Results of kidnapping a robot to 9 different positions on the field in experimental setup 1 (third column is the average of 5 experiments per location)

- Case 2: Observer robot stopped and the ball is moving while the ball is in the FOV of both robots. Robot kidnapped once in this case.

- Case 3: Both observer robot and the ball stopped when the ball is in the FOV of both robots. Robot kidnapped twice in this case.

Figure 4.5: Real robot view for one of the layout locations: the robot on the left is the robot that informs the correct ball position to the other, while the robot on the right is the kidnapped robot which uses that information, in location 4 of Figure 4.4.

- Case 4: Both observer robot and the ball moving when the ball moves away from the FOV of the kidnapped robot during the time of kidnapping and then later reappears in its FOV. Robot kidnapped once in this case.

The rest of the details for this setup is similar to experimental setup 1. Results of this experiment is presented in Table 4.2 and can also be visualized in the video[1] accompanying this chapter.

Case	Situation	Result	Iterations to converge
1	1	Success	17
	2	Success	26
2	1	Failure	18
3	1	Success	15
	2	Success	19
4	1	Success	61

Table 4.2: Results of kidnapping as explained in the experimental setup 2.

[1]Video of the experiment: Application of the Algorithm 4.2 on 2 SocRob robots.
http://users.isr.ist.utl.pt/∼aahmad/PhDThesisVideos/Chap_4_MCRL/PF_MCRL.mpg

4.3.3 Results and Discussion

Results of experiments in setup 1 are shown in Table 4.1. In the table, successes correspond to the number of experiments where the robot could regain its correct pose after kidnapping occurred. Iterations to converge refer to the mean value of iterations required by the algorithm to converge after a kidnapping, over 5 experiments for a given kidnapping location. Note that one prediction and one update iteration of MCL take approximately 0.1s each, therefore the mean value of iterations should be multiplied by 0.2s to have an idea of the time taken by the algorithm in each case. Overall, the algorithm performed quite well in real situations, including cases where we kidnapped the robot to a position where its distance to the ball remained the same. Some field locations are clearly more demanding than others (e.g., 5, 6, 7) causing the robot to fail to regain its posture in one of the tests, possibly due to perceptual aliasing relatively to other field positions.

In the experimental setup 2, the robot successfully recovers and re-localizes itself in 5 situations (3 Cases) and fails in 1. The number of iterations performed by the algorithm to converge are presented in Table 4.2. In case 4 of this setup, the robot performs a very high number of iterations to converge mainly due to the absence of the ball from kidnapped robot's FOV for a while before it comes in the FOV of both robots.

In all the sets of experiments, communication delay between the robots was consistently monitored during the run-time of the algorithm. Older data ($>$ 2seconds) was discarded. A chunk of iterations performed by the robot to converge to the right posture is attributed to this communication delay.

4.4 Summary

In this chapter we presented a modified MCL algorithm for cooperative localization of robots from a team, where an object visually observed by all the team members involved in the cooperative localization is used. The algorithm takes advantage of the information on the visually shared object, provided by teammates, to modify the particle reset step when a robot determines it is lost (e.g., because it was kidnapped). The algorithm was applied to real robots in RoboCup Soccer MSL with considerable success.

The major issue with this approach is the situation which can arise due to false positive identification of the shared object. A proper approach to solve it would be to use a fused information of the shared object, where the fusion algorithm can discard false positives detected by teammates. Secondly, a fast moving ball creates larger uncertainty about its position which also affects the robustness of the approach, presented in this chapter, to

some extent. These are among the problems addressed in later chapters where a unified multi-robot localization and object tracking approach is taken.

Ongoing work will include testing the algorithm in more demanding situation, such as during actual games, with the robots continuously moving. Also, Algorithm 4.2 could further benefit by modifying the original MCL in a way such that a fraction of the particles is always spread over a circle, depending on the ratio between the short-term average and long-term average of their weights, instead of checking when this ratio drops below a given threshold. Other objects, such as the teammates, can also be shared to improve cooperative localization, as long as one can determine their position and track them.

4.5 Related Publications

The algorithm presented in this chapter was initially published in the proceedings of RoboCup Symposium 2010, Singapore (submitted: February 2010, accepted: March, 2010). It was later included as a chapter in the book *RoboCup 2010: Robot Soccer World Cup XIV. Lecture Notes In Artificial Intelligence, Lecture Notes in Computer Science* [12].

Chapter 5

Cooperative Robot Localization and Target Tracking: A Unified Framework

5.1 Introduction

One of the approaches for multi-robot cooperative object tracking is to explicitly share information among the robots. This would include sharing self-localization estimates and environment observations to enrich other robot's observations but, for this purpose, one needs a well calibrated self-localization confidence measuring technique to avoid reference frame inconsistencies. Algorithm 3.1, presented in Chapter 3, deals with such a situation in an efficient manner. However, since the confidence measuring mechanisms are often based on heuristics, e.g, using the *number of effective particles* [44] as a measure of robot localization confidence, it might still result in false positives (e.g, a situation where the self-localization confidence of a robot is high in spite of it being wrongly localized). Similarly, cooperative robot localization can also be performed by explicitly sharing information regarding a mutually observed common object but would still be much dependent on the teammate robots' self-localization confidences as well as their object observation confidence measures when using the teammate's observation or estimates. We presented such a method (Algorithm 4.2) in Chapter 4 which suffers from the aforementioned dependency [12]. In a scenario where a team of robots needs to localize themselves as well as track an object, cooperative methods for doing both can benefit from using a single framework because this would eliminate the need for separate self-localization/object-observation confidence measurements. Note that this does not mean first performing cooperative object

tracking, and then using the tracked object for improving self-localization (i.e, a two step process), which some authors have explored before[18]. In a unified framework, both processes of cooperative localization of robots and cooperative estimation of the target need to be performed in a single step to avoid propagation of estimation error from one process to the other, which is inevitable in a two-step process. This idea does not seem to be explored much in the literature. In this chapter we present two novel methods for such a unified framework of multi-robot cooperative localization and object tracking in a systematic way and present mathematical formulations for both. The first method operates in an offline manner, processing the whole experiment dataset at once as a batch. Hence it can only be applied on a previously collected experimental dataset. The second method is an online algorithm. It is designed to operate in real-time on data acquired continuously by robots' sensors. Furthermore, it is important to note that both methods assume that the robots can only communicate information data among themselves. They cannot measure each others' positions directly, e.g, measuring inter-robot distances using vision. Extending the methods presented here to incorporate such information is not straightforward and would require significant adaptation, which we comment on as future work.

5.1.1 Offline Method

In the offline approach, we pose the problem as a multi robot cooperative pose graph optimization problem where the tracked object is treated as a moving landmark and its state, namely its position and velocity, is included in the parameter set to be estimated. Although the basic idea of having a moving landmark itself is not new and has occasionally appeared in the literature [48] where it has been studied to separate the static landmarks from the dynamic ones (e.g, a piece of furniture that could be moved around in the office), extending this concept to track a fast maneuvering object through a team of mobile robots and treating this object as a moving landmark along with other static landmarks for localizing each robot in the team is novel to the best of our knowledge. More specifically our approach of multi-robot moving landmark graph optimization (O-MMLG) consists of the following steps:

- Creating a graph for the problem described in the above paragraph representing the robots' poses and the moving landmark's (tracked object) positions and velocities at each time step as nodes whereas the odometry and observation measurements made by the robots as edges.
- Stacking all the odometry and observation measurements together to create a single

non-linear least squares error function.

- Invoking an optimization solver routine for which we use g^{2}o [33], an open-source C++ framework to find the minimum of the function which is the optimal configuration of the nodes that best describes all the measurements made by all the robots.

5.1.2 Online Method

The online approach is a PF-based algorithm for multi-robot unified cooperative localization and tracking (UCLT). In the online UCLT problem, the goal is to estimate the state, including all the the robots' poses and the object position, at a certain timestep, given all the control and observation measurements made by the robots only up to that timestep. Since PFs, at their core, are recursive Bayesian filters and assume the Markovian property of the state's completeness, the state at a certain timestep is computed from a belief distribution (represented by a set of particles), using the belief distribution at the immediate previous timestep (set of particles resulting from the immediate previous iteration of the PF algorithm) and the control and observation measurements obtained only at that timestep. In the solution (henceforth referred to as the PF-UCLT algorithm), presented later in Section 5.4, a particle's state hypothesis component (the other component of a particle is its weight) represents the pose of the robot running the algorithm, the poses of the teammate robots and the position of the tracked object. The algorithm's update step systematically overcomes the problem of error propagation from the target position estimate to robots' pose estimates which happens in a two-step process as mentioned previously. We also demonstrate how the PF-UCLT algorithm drastically reduces the space and time complexity of a standard PF-based solution for the online UCLT problem.

The rest of the chapter is organized as follows. We first discuss some related work in the literature which is followed by Section 5.3, where we present the graph optimization-based offline method for unified cooperative localization and object tracking. It consists of a brief overview of graph-based optimization and g^{2}o, our problem formulation of the O-MMLG and real robot experimental results along with its analysis. Section 5.4 first describes the mathematical formulation of the online UCLT problem, the PF-UCLT algorithm in detail and then presents its implementation results on the same dataset as used in the O-MMLG's case. We conclude with comments on future work in Section 5.5.

5.2 Related Work

Robot localization, object tracking and environment mapping have been areas of extensive research in mobile robotics [49]. An extensive number of techniques based on Kalman filter, particle filter and so on have been proposed for object tracking and robot localization. For unknown environments, where the robots need to perform mapping, many simultaneous localization and mapping (SLAM) approaches have been studied. In recent years multi-robot systems have received much popularity in the field of mobile robotics. Cooperative localization, mapping and cooperative target tracking in such systems has gained a lot of attraction by researchers in this area [50],[2],[18].

In the previous decade several techniques for data fusion and cooperation have been investigated for multi-robot systems. Durrant-Whyte et al. have done extensive research in the area of cooperative target tracking in multi robot teams. One of their works [50] introduces a decentralized particle filter (DPF) where the particles are transformed into Gaussian mixture models for communication and fusion which ensures lesser bandwidth usage while at the same time providing individual robot's particle set summaries to the whole team. Later in [51] they present a generalized formulation for DPF accounting for the correlated estimation error arising from the common past information being communicated among the robots in a team. In [18] the authors present a novel approach for cooperative target tracking and localization by using visual relations of static objects and moving target as a means of tracking the object locally in a non-robot-centric frame, which when communicated to other robots is independent of the localization uncertainty however the method gets constrained to a small environment where most of the robots need to perceive the static objects continuously at all times. Furthermore, it is a two-step approach in which the errors of target estimation are propagated to self-localization of the robots, although not recursively since target observations are in non-robot-centric frames. The usage of non-robot-centric frames can sometimes be difficult, e.g., large outdoor scenarios with few observable landmarks at a given instant, robot-centric observations are necessary. In such a case, using a two-step approach, like in [18], would then lead to recursive propagation of errors .

Scenarios in multi robot systems where mapping is required, several techniques in the form of Cooperative SLAM (C-SLAM) [3] have been proposed. A recent work by B. Kim et al. [52] introduces graph SLAM for cooperative mapping and in [53] H. Wang et al. put forward a mathematical proof for the existence of optimal pose configurations for pose graphs consisting of multiple robot pose nodes. In [48] the authors propose an extension for a single robot SLAM to include movable objects and factor them out from the map

where as in our O-MMLG method we consider a fast maneuvering object, treated as a moving landmark, being tracked cooperatively by multiple robots formulated as a graph optimization problem.

5.3 Graph Optimization-based Approach: An Offline Method

5.3.1 Graph Optimization and g^{2}o

In a graph-based optimization approach, the problem is represented as a graph where the nodes consist of the set of parameter vectors to be estimated, e.g., poses of the robot, positions of the landmarks and so on while the edges connecting these nodes represent raw sensor measurements (robot observations), more specifically a probability distribution over the relative node locations conditioned by their mutual observations [33]. Assuming these observations to be affected by Gaussian noise and a known data association, the goal of graph optimization is to find the configuration of nodes that maximizes the likelihood of all the observations which are represented as edges. In order to to so, a nonlinear error function is obtained representing the whole graph, which is then solved as a constrained optimization problem or more specifically as a nonlinear least squares problem.

g^{2}o is a general framework developed in [33] for performing the optimization of nonlinear least squares problem that can be represented as a graph. A generalized mathematical description of the graph optimization problem formulation, which can be applied directly to a single robot with static landmarks, and the g^{2}o framework is presented in [54] and [33] respectively. In the next section we use this as a basis for developing the mathematical formulation of O-MMLG.

5.3.2 Multi-Robot Moving Landmark Graph Optimization (O-MMLG)

In this section, we follow a notational scheme similar to that of [33] to provide a mathematical description of the O-MMLG problem in terms of least square minimization of a sparse system. Let N be the number of robots tracking O objects, treated as a moving landmarks, for T time steps in an environment consisting of L static and known landmarks. The n^{th} robot is denoted by r_n. Without loss of generality, the formulation could easily be extended for unknown landmarks, given known data associations.

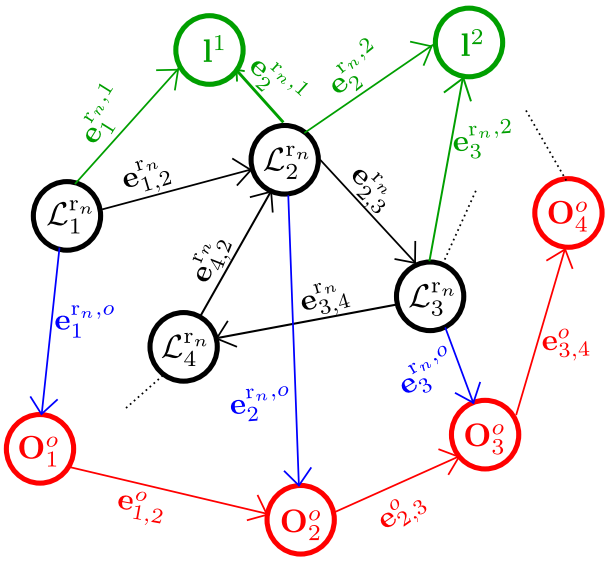


Figure 5.1: An example of the pose-graph representation of the O-MMLG depicting the robot r_n in the team, the o^{th} object (moving landmark), static and known landmarks numbered $\mathbf{l}^1$ and $\mathbf{l}^2$ and the edges connecting all these nodes. The notations are as described in Section 5.3.2.

Let $\mathcal{L}_t^{\mathrm{r}_n}$ be the pose of the robot r_n in the team at the t^{th} timestep, $\mathbf{O}_t^o$ the position and velocity of the o^{th} object at timestep t and $\mathbf{l}^l$ the position of the l^{th} static and known landmark. Let $\mathbf{z}_{t,t+1}^{\mathrm{r}_n}$ and $\mathbf{\Omega}_{t,t+1}^{\mathrm{r}_n}$ be the mean and information matrix of a virtual measurement between the pose at timestep t and $t+1$ of the robot r_n. This can be either coming from odometry or local matching algorithms. Let $\hat{\mathbf{z}}^{\mathrm{r}_n}(\mathcal{L}_t^{\mathrm{r}_n}, \mathcal{L}_{t+1}^{\mathrm{r}_n})$ be the prediction of the measurement between them. the error term described by the edge $\mathbf{e}_{t,t+1}^{\mathrm{r}_n}$ is:

$$\mathbf{e}_{t,t+1}^{\mathrm{r}_n} = \mathbf{z}_{t,t+1}^{\mathrm{r}_n} - \hat{\mathbf{z}}^{\mathrm{r}_n}(\mathcal{L}_t^{\mathrm{r}_n}, \mathcal{L}_{t+1}^{\mathrm{r}_n}). \tag{5.1}$$

Let $\mathbf{z}_t^{\mathrm{r}_n,l}$ and $\mathbf{\Omega}_t^{\mathrm{r}_n,l}$ be the mean and information matrix of a virtual measurement of the landmark l from the robot r_n at timestep t. Let $\hat{\mathbf{z}}^l(\mathcal{L}_t^{\mathrm{r}_n}, \mathbf{l}^l)$ be the prediction of the measurement, the error term of the edge is $\mathbf{e}_t^{\mathrm{r}_n,l}$:

$$\mathbf{e}_t^{\mathrm{r}_n,l} = \mathbf{z}_t^{\mathrm{r}_n,l} - \hat{\mathbf{z}}^l(\mathcal{L}_t^{\mathrm{r}_n}, \mathbf{l}^l). \tag{5.2}$$

Similarly, the error term between the object o's virtual and predicted measurements made by the robot r_n at timestep t is:

$$\mathbf{e}_t^{\mathrm{r}_n,o} = \mathbf{z}_t^{\mathrm{r}_n,o} - \hat{\mathbf{z}}^o(\mathcal{L}_t^{\mathrm{r}_n}, \mathbf{O}_t^o). \tag{5.3}$$

The motion of the objects is modeled using a constant velocity motion model with random acceleration, leading to the following term for the edge between consecutive object

positions, $\mathbf{e}^o_{t,t+1}$:

$$\mathbf{e}^o_{t,t+1} = \mathbf{O}^o_{t+1} - \mathbf{A}\mathbf{O}^o_t - \nu, \tag{5.4}$$

where $\mathbf{A}$ is the matrix modeling discrete-time constant velocity and ν is a zero mean error with information matrix $\mathbf{\Omega}^o_{t,t+1}$.

Let now $\mathbf{X}$ be the vector obtained by stacking all the variables which consists of the pose $\mathcal{L}^{\mathrm{r}_n}_t$ of all the robots $\mathrm{r}_1, .., \mathrm{r}_N$ and the object position $\mathbf{O}^o_t$ of all the tracked objects (moving landmarks) 1 to O, both from timestep $t = 1$ to $t = T$. The solution of the O-MMLG is the $\mathbf{x}$ that minimize the following function

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} \mathbf{F}(\mathbf{X}) \tag{5.5}$$

$$\begin{aligned}\mathbf{F}(\mathbf{X}) = &\sum_{n=1}^{N}\sum_{t=1}^{T-1} {\mathbf{e}^{\mathrm{r}_n}_{t,t+1}}^\top \mathbf{\Omega}^{\mathrm{r}_n}_{t,t+1}\mathbf{e}^{\mathrm{r}_n}_{t,t+1} + \\ &+ \sum_{n=1}^{N}\sum_{l,t\in\mathcal{S}_l} {\mathbf{e}^{\mathrm{r}_n,l}_t}^\top \mathbf{\Omega}^{\mathrm{r}_n,l}_t\mathbf{e}^{\mathrm{r}_n,l}_t + \\ &+ \sum_{n=1}^{N}\sum_{o,t\in\mathcal{S}_o} \mathbf{e}^{\mathrm{r}_n,o\top}_t \mathbf{\Omega}^{\mathrm{r}_n,o}_t\mathbf{e}^{\mathrm{r}_n,o}_t + \\ &+ \sum_{o=1}^{O}\sum_{t=1}^{T-1} {\mathbf{e}^o_{t,t+1}}^\top \mathbf{\Omega}^o_{t,t+1}\mathbf{e}^o_{t,t+1},\end{aligned} \tag{5.6}$$

where $\mathcal{S}_l$ and $\mathcal{S}_o$ are respectively the set of all the observations between any robot and any static landmark or any tracked object (moving landmark). See Figure 5.1 for a visual explanation on how the graph is built. Green nodes represent the static and known landmark locations, black nodes represent the robot poses and red nodes represent the tracked object's positions. Red arrows indicate the motion of the individual tracked objects. Robot observations of static landmarks and moving objects are shown respectively with green and blue arrows.

The formulation of the final objective function (5.6) is similar to the one obtained in [54] and [33]. Hence following their procedure, we can approximate it using Taylor expansion around an initial guess for $\mathbf{x}$ and applying error minimization via iterative local linearizations or least squares on a manifold.

5.3.3 Experiments and Results

5.3.3.1 Testbed and Experimental Scenario

We applied the O-MMLG, proposed here, to the robot soccer scenario. Our testbed is the RoboCup Middle Sized League (MSL) (see Appendix A for a detailed description of the testbed). The primary requisites for a team of robots playing soccer are : i) individual robot localization and ii) continuous tracking of the soccer ball. The localization and/or ball tracking failures in such a scenario owe much to the field size and symmetry, limited sensor range, occlusions as well as dynamic and fast movements of the robots and/or the ball. This makes the robot soccer scenario an interesting and suitable testbed for evaluating the performance of the O-MMLG. From the datasets collected on the testbed (see Appendix A for the details of the dataset), we used the one, containing 4 robots' data. From the logs of this dataset, only 6 landmarks out of 10 were used in this experiment.

The O-MMLG was then implemented on the odometry and observation data logs and the results were evaluated against the the ground truth system (GTS) estimates as explained in Appendix B. The GTS estimates all the robots' and the ball's global positions (X and Y). It does not estimate the robot's orientation. Hence the comparisons with the GT consists of comparing the positions of the robots and the ball but not the orientation of the robots. In order to compare the O-MMLG's results, we implemented an extended Kalman filter (EKF) for cooperative localization and object tracking. In the EKF the state vector estimated at any time step t consists of all the robots poses $(\mathcal{L}_t^{\mathrm{r}1}, ..., \mathcal{L}_t^{\mathrm{r}N})^\top \in \mathbb{R}^{3\times \mathrm{N}}$ and the ball's 2D position and velocity $\mathbf{O}_t^{\mathrm{ball}} \in \mathbb{R}^4$. In the prediction step we used for the the odometry information for the robot poses and for the ball positions we implemented a constant velocity and zero mean white Gaussian acceleration noise model [55]. In the update step, observations of static landmarks and the ball from each robot are synchronized to obtain a joint measurement vector (for a detailed description of the EKF formulation, please refer Appendix C).

In sub-subsection 5.3.3.2 we present the results of the O-MMLG compared with the EKF-based approach implemented on the data logs of the 2 robots OMNI1 and OMNI2 for localization and ball tracking. The aim of this experiment is to present a proof of concept of the O-MMLG and the higher accuracy achieved by it compared with the EKF-based approach.

In sub-subsection 5.3.3.3 we extended the implementation of both the approaches to all the 4 robots OMNI1 to OMNI4. The aim here is to show the scalability of our approach while still achieving a higher degree of accuracy over the EKF-based approach.

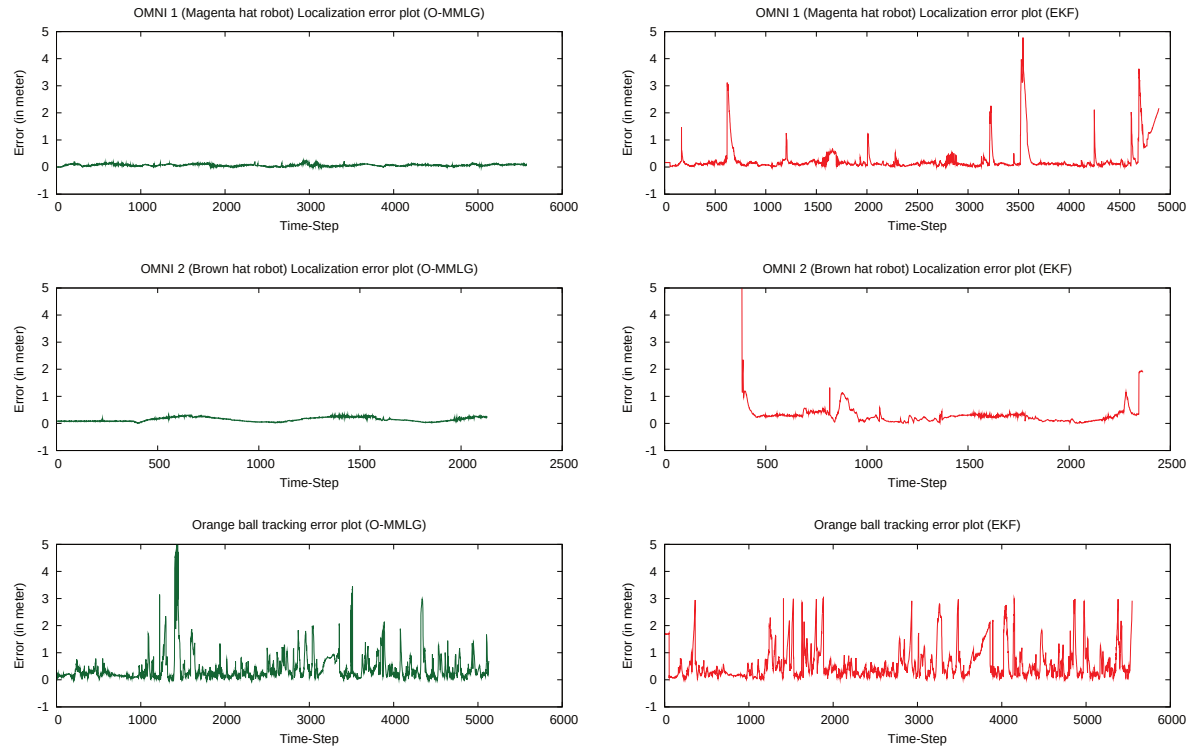


Figure 5.2: 2 robots experiment : The X-axis in these plots represent the number of time-steps between two consecutive GTS frames. If for a particular robot or the ball the GTS could not estimate the GT, error w.r.t. it is not computed at that time step and hence omitted from the plots. The Y-axis represents the position error which is the Euclidean distance between the GT position estimates and the O-MMLG (left) or EKF (right) estimates of the robot and the orange ball positions.

In the video[1] accompanying this section we present the 4 robots experiment's footage composed by concatenating the stream of images from one of the GTS's cameras. Each robot has a uniquely colored hat placed on top of it (OMNI1:Magenta, OMNI2:Brown, OMNI3:Red and OMNI4:Blue). The GTS uses these hats for detecting the positions of the robots. The GT estimates by the GTS are marked in the video frames with an overlaid black circle around the robot's hat and the orange ball except for the instances during which the robots' markers or the ball gets occluded from the GTS and it could not detect the robots' or the ball's positions. The O-MMLG estimates of the robot positions are

[1]Video link of the O-MMLG 4 robots experiment.
http://users.isr.ist.utl.pt/∼aahmad/PhDThesisVideos/Chap_5_O-MMLG/O_MMLG.mpg

marked in the video frames by overlaying a circle of the same color as that of the robot's hat. These circles are centred at the O-MMLG's position estimate for the particular robot and are placed at the same height as of the robot's hat from the ground level to facilitate easy visual comparison. The robot's orientation estimate by the O-MMLG is marked by a radial line from the circle's center. The O-MMLG's orange ball position estimates are marked by an orange circle overlaid on the video frames.

5.3.3.2 2 Robots Experiment

In this sub-subsection the results of the O-MMLG and EKF-based approaches implemented on the data logs of OMNI1 and OMNI2 are presented. The plots in the left column of Figure 5.2 display the error of both robots' positions and the ball positions as estimated by the O-MMLG w.r.t. the GT. The axes in the plots are explained in Figure 5.2's caption. The plots in the right column of Figure 5.2 display the errors of the EKF-based approach's estimates w.r.t. its GT counterparts. Table 5.1 shows the statistical estimates of the error plots presented in Figure 5.2.

From Figure 5.2 and Table 5.1 we infer that the O-MMLG is able to reduce the mean error in position estimates for OMNI1 by a factor of 4, for OMNI2 by a factor of 2 and for the ball by a factor of 1.3 compared to the EKF-based approach.

	O-MMLG			**Cooperative EKF**		
	Mean (m)	Median (m)	Variance (m^2)	Mean (m)	Median (m)	Variance (m^2)
OMNI1	0.069	0.066	0.001	0.274	0.123	0.272
OMNI2	0.147	0.127	0.006	0.294	0.261	0.077
Ball	0.426	0.240	0.314	0.583	0.356	0.357

Table 5.1: Statistical estimates of the 2 Robots experiment results.

5.3.3.3 4 Robots Experiment

Figure 5.3 presents the error plots of all the four robots and the orange ball for the O-MMLG in the left column (green-colored plots) of the figure and the EKF-based approach in the right column (red-colored plots). In the EKF-based approach, the robots (most notably OMNI4) often tend to lose their correct localization due to noisy observation and odometry measurements. However, as soon as better measurements arrive, the EKF recovers the robots' position estimates and the error gets reduced. This effect is mitigated

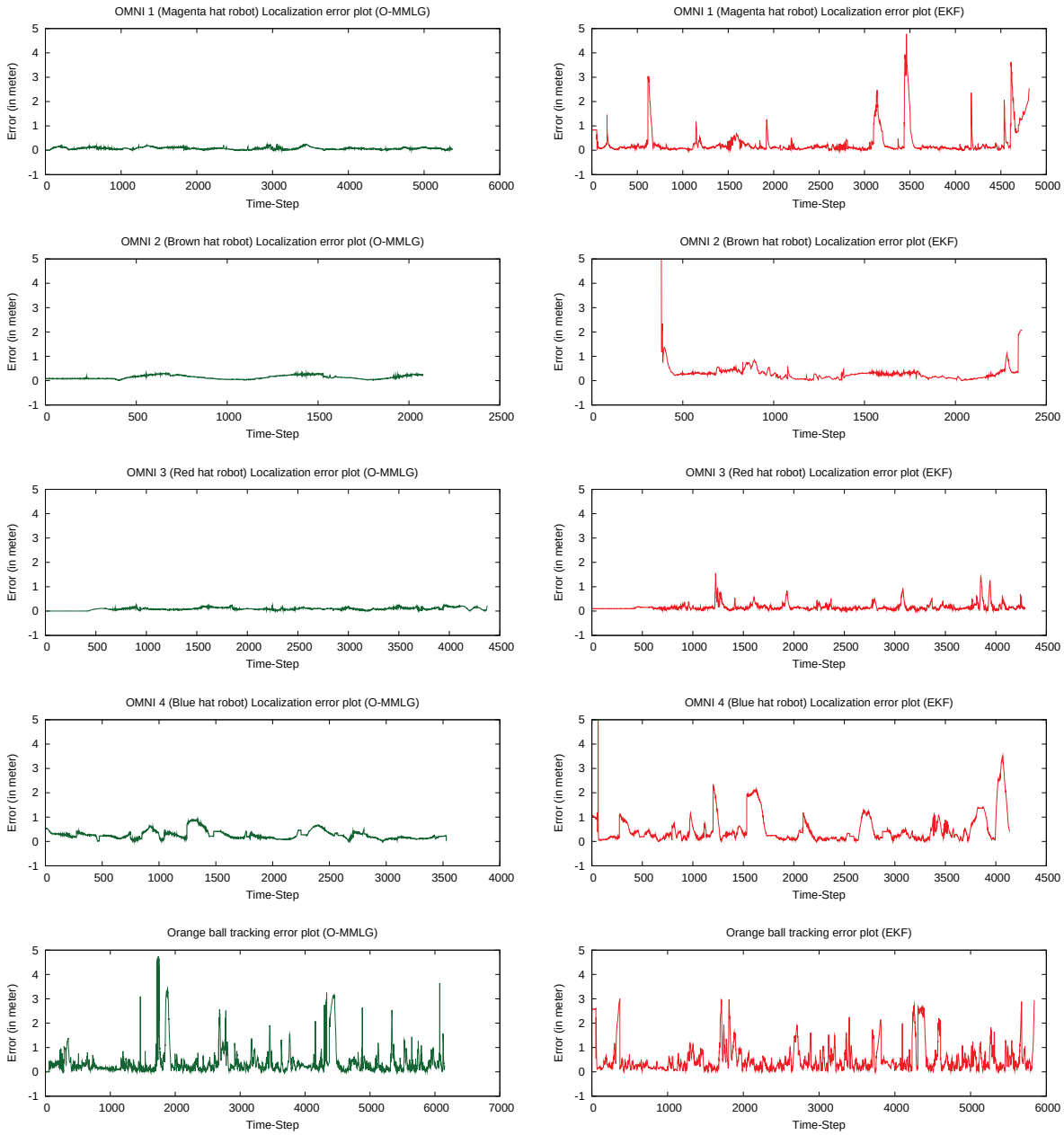


Figure 5.3: 4 robots experiment : The X-axis in these plots represent the number of time-steps between two consecutive GTS frames. If for a particular robot or the ball the GTS could not estimate the GT, error w.r.t. it is not computed at that time step and hence omitted from the plots. The Y-axis represents the position error which is the Euclidean distance between the GT position estimates and the O-MMLG (left) or EKF (right) estimates of the robot and the orange ball positions.

	O-MMLG			Cooperative EKF		
	Mean (m)	Median (m)	Variance (m^2)	Mean (m)	Median (m)	Variance (m^2)
OMNI1	0.077	0.070	0.002	0.298	0.118	0.295
OMNI2	0.141	0.120	0.006	0.296	0.265	0.075
OMNI3	0.090	0.086	0.003	0.166	0.124	0.024
OMNI4	0.267	0.221	0.030	0.493	0.276	0.327
Ball	0.399	0.226	0.327	0.528	0.315	0.334

Table 5.2: Statistical estimates of the 4 Robots experiment results.

in O-MMLG thanks to the iterative relinearizations. This can be observed in the EKF-based approach's error plots of these robots in Figure 5.3, when compared to O-MMLG. The latter is able to maintain an acceptable estimate for all these robots even in the case of highly noisy measurements and observations. From Figure 5.3 and Table 5.2 we infer that O-MMLG outperforms the EKF-based approach by a factor of 3.8 for OMNI1, 2.1 for OMNI2, 1.8 for OMNI3, 1.8 for OMNI4 and 1.3 for the orange ball.

5.3.3.4 Computation Time Comparison

	O-MMLG	EKF
2 Robot Exp.	13.75s	13.38s
4 Robot Exp.	19.38s	137.95s

Table 5.3: Comparison of the total computation time taken by the O-MMLG and the cooperative EKF approach on the full dataset.

We implemented both the O-MMLG and the EKF based approach on the same machine (Quad Core Intel(R) Core(TM) i5 CPU 750 @ 2.67GHz, 8GB RAM) for a fair comparison of the approaches' computation time. Table 5.3 presents the total time taken by both implementations on the same dataset. For the O-MMLG, this reflects the total time taken by the g^2o's optimization process for the whole graph of the dataset. The graph consists of 50000 nodes in the case of 4 robots and 30000 nodes in the case of 2 robots. For the EKF based approach it reflects the total time taken to iterate once over the whole dataset. The O-MMLG required 87 iterations for the 30000 node graph and 60 for the 50000 node graph to converge. The time required by the O-MMLG grows linearly with

respect to the number of robots. On the other hand, in the EKF based approach, the required computation time grows quadratically. Compared to the EKF-based approach's slow execution speed and higher degree of inaccuracy, O-MMLG's scalability and accuracy makes it a choice of higher value for cooperative localization and object tracking.

5.4 Particle Filter-Based Approach: An Online method

In this section we present a PF-based algorithm to solve the problem of online unified cooperative localization and tracking (UCLT). Henceforth we refer to this solution as the PF-UCLT algorithm. It is designed in a decentralized manner, such that each robot runs its own instance of the algorithm where it receives all the measurements made by its own sensors and by the teammate robots' sensors, assumed to be transmitted to it through a wireless communication system. Since the sensor observation models are usually represented in a compact way by a mean vector and a covariance matrix, only a small amount of information is communicated among the robots, instead of the whole particle set. The PF-UCLT algorithm's particles represent the state (position and orientation) of the robot running the algorithm, the states of all the teammate robots and the state (position) of the tracked object. The crux of this algorithm lies in the particles' weight generation step (the update step where the observation measurements are incorporated). Here we exploit the properties of conditional and/or total independence of the involved variables (e.g, the control/sensor measurements and the individual robot/objects' state) in the problem. These properties are enumerated further in this section, as we proceed with the online UCLT problem's mathematical formulation. Since all the measurements (static landmark observation for the robots' localization and the tracked object observation made by the robots) made by the sensors at a given timestep cumulatively influence the weight of a particle, it makes sure that the two-step process, of estimating the target first and then using it to localize the observing robots, is avoided and hence there is no propagation of error from the tracked object's estimate to the robots' pose estimates or the other way around.

5.4.1 Online UCLT Problem Formulation

In this subsection we formulate the online UCLT problem, described above, using a recursive Bayesian filter. Let there be N robots $\mathrm{r}_1, .., \mathrm{r}_N$ in a team tracking an object $\mathbf{O}$ in an environment consisting of L known and static landmarks represented as a set $\mathbf{L}_{\mathrm{map}}$. The

state (pose in the world frame) of the robot r_n is given by $\mathcal{L}_t^{\mathrm{r}_n} = [x_t^{\mathrm{r}_n}\ y_t^{\mathrm{r}_n}\ \theta_t^{\mathrm{r}_n}]^\top$ and the state (3D-position in the world frame) of the tracked moving object is given by $\mathbf{O}_t = [x_t^{\mathbf{o}}\ y_t^{\mathbf{o}}\ z_t^{\mathbf{o}}]^\top$ at the t^{th} timestep. The velocity of the object is measured by a velocity sensor, implemented separately from the PF-UCLT algorithm. It is given by $v_t^{\mathbf{o}} = \{\mathbf{v}_t^{\mathbf{o}}, \mathbf{\Sigma}_t^{\mathbf{o}}\}$ where $\mathbf{v}_t^{\mathbf{o}} = [\mathbf{v}_{x_t}^{\mathbf{o}}\ \mathbf{v}_{y_t}^{\mathbf{o}}\ \mathbf{v}_{z_t}^{\mathbf{o}}]^\top$ is the mean velocity vector at timestep t and $\mathbf{\Sigma}_t^{\mathbf{o}}$ is a zero mean white Gaussian acceleration noise. The 2D-position of the l^{th} known and static landmark is given $\mathbf{l}^l = [\mathbf{l}_x^l\ \mathbf{l}_y^l]^\top$. The landmarks are assumed to be fixed on the ground plane on which the robots move.

The odometry measurement made by the robot r_n at the t^{th} timestep is given by the vector $\mathbf{u}_t^{\mathrm{r}_n}$ and an associated noise with zero mean and covariance matrix $\mathbf{R}_t^{\mathrm{r}_n}$. The static landmark observation measurement of the l^{th} landmark made by the robot r_n in its local frame at the t^{th} timestep is given by the vector $\mathbf{z}_t^{\mathrm{r}_n,l}$ and an associated noise with zero mean and covariance matrix $\mathbf{Q}_t^{\mathrm{r}_n,l}$. Similarly, the moving object $\mathbf{O}$'s observation measurement made by the robot r_n in its local frame at the t^{th} timestep is given by the vector $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$ and an associated noise with zero mean and covariance matrix $\mathbf{\Sigma}_t^{\mathrm{r}_n,\mathbf{o}}$.

We now define $\mathbf{x}_t$ as the full state vector being estimated by stacking all individual states at the t^{th} timestep as follows.

$$\mathbf{x}_t = \begin{bmatrix} \mathcal{L}_t^{\mathrm{r}_1\top}\ \ldots\ \mathcal{L}_t^{\mathrm{r}_N\top}\ \mathbf{O}_t{}^\top \end{bmatrix}^\top \tag{5.7}$$

$\mathbf{u}_t$ is obtained by stacking all control measurements (robot odometry and the tracked object's velocity) available at the t^{th} timestep as follows.

$$\mathbf{u}_t = \begin{bmatrix} \mathbf{u}_t^{\mathrm{r}_1\top}\ \ldots\ \mathbf{u}_t^{\mathrm{r}_N\top}\ \mathbf{v}_t^{\mathbf{o}\top} \end{bmatrix}^\top \tag{5.8}$$

$\mathbf{z}_t$ is obtained by stacking all the observation measurements available at the t^{th} timestep as follows.

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{z}_t^{\mathrm{r}_1,1\top}\ \ldots\ \mathbf{z}_t^{\mathrm{r}_1,L\top}\ \ \ldots\ \ \mathbf{z}_t^{\mathrm{r}_N,1\top}\ \ldots\ \mathbf{z}_t^{\mathrm{r}_N,L\top}\ \ \mathbf{z}_t^{\mathrm{r}_1,\mathbf{o}\top}\ \ldots\ \mathbf{z}_t^{\mathrm{r}_N,\mathbf{o}\top} \end{bmatrix}^\top \tag{5.9}$$

The online UCLT problem seeks to estimate $bel(\mathbf{x}_t)$, the posterior belief of the state $\mathbf{x}_t$ at the t^{th} timestep, given all the measurement data up to that timestep. This is given by a probability distribution over the state space $bel(\mathbf{x}_t) = p(\mathbf{x}_t\ |\mathbf{z}_{1:t}\ , \mathbf{u}_{1:t})$, conditioned on the available measurement data. The recursive Bayesian filter equation, under the Markovian assumption of the state's completeness [43], for the problem is as follows:

$$p(\mathbf{x}_t\ |\mathbf{z}_{1:t}\ , \mathbf{u}_{1:t}) \overset{Bayes}{=} \eta\ p(\mathbf{z}_t\ |\mathbf{x}_t,\ \mathbf{z}_{1:t-1}\ , \mathbf{u}_{1:t})\ p(\mathbf{x}_t\ |\ \mathbf{z}_{1:t-1}\ , \mathbf{u}_{1:t}) \tag{5.10}$$

$$\overset{Markov}{=} \eta\ p(\mathbf{z}_t\ |\mathbf{x}_t)\ p(\mathbf{x}_t\ |\ \mathbf{z}_{1:t-1}\ , \mathbf{u}_{1:t})$$
$$\overset{\text{TPT}}{=} \eta\ p(\mathbf{z}_t\ |\mathbf{x}_t) \int p(\mathbf{x}_t\ |\ \mathbf{x}_{t-1},\ \mathbf{z}_{1:t-1},\ \mathbf{u}_{1:t})\ p(\mathbf{x}_{t-1}\ |\ \mathbf{z}_{1:t-1}\ , \mathbf{u}_{1:t})\ d\mathbf{x}_{t-1}$$
$$\overset{\text{Markov}}{=} \eta\ p(\mathbf{z}_t\ |\mathbf{x}_t) \int p(\mathbf{x}_t\ |\ \mathbf{x}_{t-1},\ \mathbf{u}_t)\ p(\mathbf{x}_{t-1}\ |\ \mathbf{z}_{1:t-1},\ \mathbf{u}_{1:t-1})\ d\mathbf{x}_{t-1}$$

where η is the proportionality constant and TPT stands for the "*total probability theorem*".

5.4.2 PF-UCLT Algorithm: A Solution to The Online UCLT Problem

In this subsection we first discuss a standard PF solution to the recursive Bayesian formulation of the online UCLT problem given by (5.10). We show how the weight generation step of a traditional PF solution could cause it to have an exponentially high computational complexity. Subsequently, the PF-UCLT algorithm is presented which modifies the PF's weight generation step in a way such that the computational complexity is substantially reduced.

PF approximates $bel(\mathbf{x}_t)$ by a set of M particles $\mathcal{X}_t \triangleq \{\langle \mathbf{x}_t^{[m]}, w_t^{[m]} \rangle\}_{m=1}^{M}$. The components of any m^{th} particle $\langle \mathbf{x}_t^{[m]}, w_t^{[m]} \rangle$ are $\mathbf{x}_t^{[m]}$, a hypothesis about the state $\mathbf{x}_t$ and an associated weight $w_t^{[m]}$ which, ideally, should be proportional to its Bayes filter posterior $bel(\mathbf{x}_t)$ [43]. In a standard PF implementation to solve the online UCLT problem formulated in (5.10), the goal is to estimate the particle set $\mathcal{X}_t$, given the particle set $\mathcal{X}_{t-1}$, all the control measurements $\mathbf{u}_t$, all the observation measurements $\mathbf{z}_t$ and the static landmarks' known map $\mathbf{L}_{\text{map}}$. The standard PF solution is described in the following four steps.

Step 1: A temporary particle set $\bar{\mathcal{X}}_t \triangleq \{\langle \bar{\mathbf{x}}_t^{[m]}, \bar{w}_t^{[m]} \rangle\}_{m=1}^{M}$ is initialized with null values (zeros).

Step 2: The state hypothesis component $\bar{\mathbf{x}}_t^{[m]}$, of an m^{th} temporary particle is obtained by the sampling process in (5.11) which incorporates all the control measurements:

$$\bar{\mathbf{x}}_t^{[m]} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{[m]}, \mathbf{u}_t) \tag{5.11}$$

Step 3: The weight component $\bar{w}_t^{[m]}$, of an m^{th} temporary particle is obtained as per (5.12) which incorporates all the observation measurements.

$$\bar{w}_t^{[m]} \propto p(\mathbf{z}_t | \bar{\mathbf{x}}_t^{[m]}, \mathbf{L}_{\text{map}}) \tag{5.12}$$

Step 4: After Steps 2 and 3 are performed for all the M particles, the resulting temporary set $\bar{\mathcal{X}}_t$ is resampled to obtain $\mathcal{X}_t$. Resampling is the process where M particles are drawn with replacement and the probability of drawing a particle is proportional to its weight. This can be done using various resampling techniques existing in the literature, e.g, low variance sampler [43]. Lastly, the state estimate $\mathbf{x}_t$ is obtained from the resampled particle set $\mathcal{X}_t$.

The performance of a PF is heavily dependent on the number of particles [56] [57] [58] [59] [60] [61]. In practice, to achieve a good approximation of the posterior belief $bel(\mathbf{x}_t)$ by a PF-based method and to not quickly fall into the particle deprivation problem [43], the number of particles depends exponentially [56] [60] on the dimension of the state space represented by a particle. The particle deprivation problem, as described in [43], refers to a situation where none of the particles are in the vicinity of the correct state. This is more likely to happen as the dimension of the state space grows. In [56], Quang et al. formally prove that the PF error increases exponentially with the estimated state's dimension and therefore, to maintain a given accuracy, the number of particles used in a PF must increase exponentially with the state's dimension. We further discuss the role of state's dimensionality in the context of the UCLT problem.

Let us assume that in case of a single robot localization problem, where the robot moves on a plane causing the state space dimension to be 3 (robot's 2D position and orientation), the required number of particles for an acceptable accuracy by a PF-based localization method is M (in practice, M is usually tractably small, i.e, M in the order of thousands results in an acceptable accuracy and computational speed). However, when the state space consists of N robots' poses tracking O objects' 3D positions, the number of particles required must be $M^{(N+O)}$ for a similar accuracy to that obtained by M particles in case of a single-robot localization. This renders the usage of a standard PF implementation very inefficient for the UCLT problem. Even with $N = 2$ and $O = 1$, which is the minimum requirement for a UCLT scenario, the required number of particles will be in the order of millions.

In the PF-UCLT algorithm (Algorithm 5.1), we overcome this issue by utilizing the properties of conditional and total independence of some of the involved variables and accordingly modify the weight association process of the temporary particles (Step 3 of the standard PF solution described above), eventually causing the required number of particles (and hence the space complexity of the PF-UCLT algorithm) to grow only linearly with $N + O$, which otherwise would grow exponentially with respect to that. More specifically, if M is the required number of particles to maintain a given accuracy for a PF-based

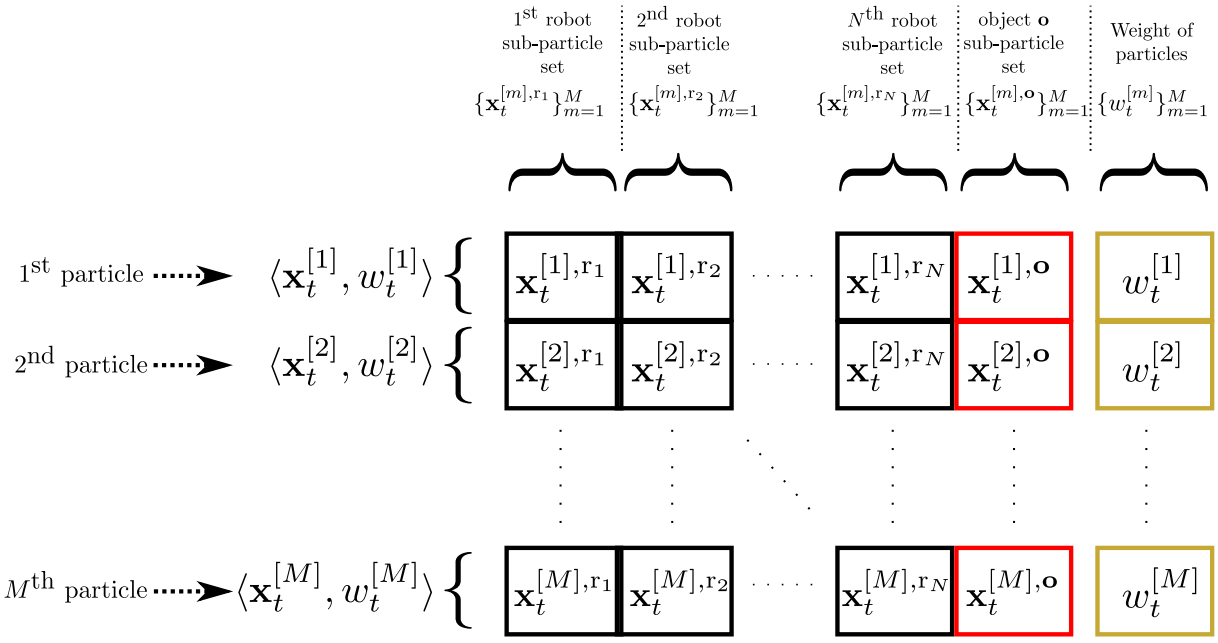


Figure 5.4: Tabular representation of the particles, sub-particles and the associated notations defined in Section 5.4 and used in Algorithm 5.1 and subsequently.

single robot localization, then for the UCLT problem with N robots and O objects, the PF-UCLT algorithm would require only $(N+O)M$ particles instead of $M^{(N+O)}$ particles. However, the time complexity, as described further, will reduce from exponential to linear only w.r.t N and not w.r.t O. Despite this limitation, the proposed solution is able to perform efficiently for realtime applications when the number of tracked objects O is low.

PF-UCLT Algorithm Description

The Algorithm 5.1 and its description here considers the number of tracked objects $O = 1$ for the sake of notational simplicity, with no loss of generality

Before describing the PF-UCLT algorithm, it is important to introduce the concept of sub-particles used throughout the rest of the chapter. Since the state hypothesis component $\mathbf{x}_t^{[m]}$ of an m^{th} particle $\langle \mathbf{x}_t^{[m]}, w_t^{[m]} \rangle$ is composed of the states of all robots $\mathrm{r}_1, .., \mathrm{r}_N$ and the tracked object $\mathbf{O}$ at the timestep t, we define a particle to be an $(N+2)$-tuple as follows:

$$\underbrace{\langle \mathbf{x}_t^{[m]}, w_t^{[m]} \rangle}_{m^{\text{th}}\text{particle}} \overset{\text{def.}}{=} \underbrace{\langle \mathbf{x}_t^{[m],\mathrm{r}_1}, ..., \mathbf{x}_t^{[m],\mathrm{r}_N}, \mathbf{x}_t^{[m],\mathbf{o}}, w_t^{[m]} \rangle}_{(N+2)\text{-tuple}}, \tag{5.13}$$

where the first N elements of the $(N+2)$-tuple, henceforth designated as the robot sub-particles of the m^{th} particle, form an N-tuple $\langle \mathbf{x}_t^{[m],\mathrm{r}_1}, ..., \mathbf{x}_t^{[m],\mathrm{r}_N} \rangle$ and represent the state

hypothesis of the robots $\mathrm{r}_1, .., \mathrm{r}_N$. The $(N+1)^{\text{th}}$ element $\mathbf{x}_t^{[m],\mathbf{o}}$, henceforth designated as the object sub-particle of the m^{th} particle, represents the state hypothesis of the tracked object $\mathbf{O}$. The last $(N+2)^{\text{th}}$ element $w_t^{[m]}$ represents the weight of the full m^{th} particle. Similar sub-particle notations for temporary particles or particles at timestep $t-1$ will follow.

We now expand (5.11) and (5.12) to further facilitate Algorithm 5.1 description. Using the concept of sub-particles and the variable definitions in (5.7) and (5.8) we can expand the prediction step (5.11) as

$$\bar{\mathbf{x}}_t^{[m]} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{[m]}, \mathbf{u}_t) \tag{5.14}$$
$$\Rightarrow \langle \bar{\mathbf{x}}_t^{[m],\mathrm{r}_1}, ..., \bar{\mathbf{x}}_t^{[m],\mathrm{r}_N}, \bar{\mathbf{x}}_t^{[m],\mathbf{o}} \rangle \sim p(\mathcal{L}_t^{\mathrm{r}_1}, ..., \mathcal{L}_t^{\mathrm{r}_N},\ \mathbf{O}_t \mid \langle \mathbf{x}_{t-1}^{[m],\mathrm{r}_1}, ..., \mathbf{x}_{t-1}^{[m],\mathrm{r}_N}, \mathbf{x}_{t-1}^{[m],\mathbf{o}} \rangle, \mathbf{u}_t^{\mathrm{r}_1}, ..., \mathbf{u}_t^{\mathrm{r}_N}, \mathbf{v}_t^{\mathbf{o}})$$

Property 1: Since all the control measurements reflect individual entities' (robots' or tracked object's) measurements only and are independent from each other, the prediction steps of each sub-particle can be done separately.

Using ***Property 1***, (5.14) can be written as the following set of equations:

$$\begin{aligned}
\bar{\mathbf{x}}_t^{[m],\mathrm{r}_1} &\sim p(\mathcal{L}_t^{\mathrm{r}_1} \mid \mathbf{x}_{t-1}^{[m],\mathrm{r}_1},\ \mathbf{u}_t^{\mathrm{r}_1}) \\
&\cdot \\
&\cdot \\
\bar{\mathbf{x}}_t^{[m],\mathrm{r}_N} &\sim p(\mathcal{L}_t^{\mathrm{r}_N} \mid \mathbf{x}_{t-1}^{[m],\mathrm{r}_N},\ \mathbf{u}_t^{\mathrm{r}_N}) \\
\bar{\mathbf{x}}_t^{[m],\mathbf{o}} &\sim p(\mathbf{O}_t \mid \mathbf{x}_{t-1}^{[m],\mathbf{o}},\ \mathbf{v}_t^{\mathbf{o}})
\end{aligned} \tag{5.15}$$

Using (5.7) and (5.9), we can expand (5.12) as

$$\begin{aligned}
\bar{w}_t^{[m]} &\propto p(\mathbf{z}_t \mid \bar{\mathbf{x}}_t^{[m]}, \mathbf{L}_{\text{map}}) \\
&\propto p(\mathbf{z}_t^{\mathrm{r}_1,1}, ..., \mathbf{z}_t^{\mathrm{r}_1,L}, ..., \mathbf{z}_t^{\mathrm{r}_N,1}, ..., \mathbf{z}_t^{\mathrm{r}_N,L}, \mathbf{z}_t^{\mathrm{r}_1,\mathbf{o}}, ..., \mathbf{z}_t^{\mathrm{r}_N,\mathbf{o}} \mid \langle \bar{\mathbf{x}}_t^{[m],\mathrm{r}_1}, ..., \bar{\mathbf{x}}_t^{[m],\mathrm{r}_N}, \bar{\mathbf{x}}_t^{[m],\mathbf{o}} \rangle, \mathbf{L}_{\text{map}}) \\
&\propto \prod_{n=1}^{N} \prod_{l=1}^{L} p(\mathbf{z}_t^{\mathrm{r}_n,l} \mid \langle \bar{\mathbf{x}}_t^{[m],\mathrm{r}_1}, ..., \bar{\mathbf{x}}_t^{[m],\mathrm{r}_N}, \bar{\mathbf{x}}_t^{[m],\mathbf{o}} \rangle, \mathbf{L}_{\text{map}}) \prod_{n=1}^{N} p(\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}} \mid \langle \bar{\mathbf{x}}_t^{[m],\mathrm{r}_1}, ..., \bar{\mathbf{x}}_t^{[m],\mathrm{r}_N}, \bar{\mathbf{x}}_t^{[m],\mathbf{o}} \rangle, \mathbf{L}_{\text{map}}) \\
&\propto \prod_{n=1}^{N} \prod_{l=1}^{L} p(\mathbf{z}_t^{\mathrm{r}_n,l} \mid \bar{\mathbf{x}}_t^{[m],\mathrm{r}_n}, \mathbf{L}_{\text{map}}) \prod_{n=1}^{N} p(\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}} \mid \bar{\mathbf{x}}_t^{[m],\mathrm{r}_n}, \bar{\mathbf{x}}_t^{[m],\mathbf{o}})
\end{aligned} \tag{5.16}$$

The third line of (5.16) is derived from the second line of (5.16) using the following property of conditional independence.

Property 2: All static landmarks' observation measurements and the moving object observation measurements are independent of each other given the predicted poses of the observing robots and the predicted position of the tracked object.

The final expression of (5.16) is obtained from the third line expression of (5.16) using the property of mutual independence for some (and not all) of the involved variables. These properties are as follows:

Property 3: The observation measurement for a given static landmark made by any robot r_n is dependent only on the predicted pose of the robot r_n and the fixed position of that static landmark. It is independent of the pose of all the other robots and the position of the tracked object.

Property 4: The tracked object's observation measurement made by any robot r_n depends only on the predicted pose of the robot r_n and the predicted position of the tracked object. It is independent of the pose of all the other robots and the fixed positions of the static landmarks.

Using i) the notations defined and described earlier in this section ii) the ***Properties 1-4*** stated above and iii) the prediction and update step equations' expanded forms ((5.15) and (5.16), respectively), we now proceed to a step-wise description of the Algorithm 5.1.

The Algorithm 5.1, which is a recursive predict-update loop, requires a separate instance of it to run on each robot in the team. Further in the description we assume that Algorithm 5.1 runs on the robot r_k. The algorithm's first input is $\mathcal{X}_{t-1}$, the particle set returned from the immediate preceding iteration of the algorithm. The other inputs, as defined previously in this section, are $\mathbf{u}_t, \mathbf{z}_t, \mathbf{L}_{\text{map}}$ and r_k, all the control and observation measurements, the static landmarks positions and the robot number running the algorithm.

In lines $1-2$ of the Algorithm 5.1, the robot r_k transmits its control and observation measurements to all the other robots in the team and receives the same from every other teammate. Lines $3-4$ define temporary variables $\bar{\mathcal{X}}_t$ and $\mathcal{W}_t$, both of which are initialized with null values (zeros) and later used in the algorithm to contain intermediate step values. Variable $\bar{\mathcal{X}}_t \triangleq \{\langle \bar{\mathbf{x}}_t^{[m]}, \bar{w}_t^{[m]} \rangle\}_{m=1}^{M} \overset{\text{def.}}{=} \{\langle \bar{\mathbf{x}}_t^{[m],\mathrm{r}_1}, ..., \ \bar{\mathbf{x}}_t^{[m],\mathrm{r}_N}, \ \bar{\mathbf{x}}_t^{[m],\mathbf{o}}, \ \bar{w}_t^{[m]} \rangle\}_{m=1}^{M}$ is a temporary particle set, where any temporary particle is an $(N+2)$-tuple, similar to the one described in (5.13). Variable $\mathcal{W}_t$ is defined as a set $\mathcal{W}_t \triangleq \{\langle w_t^{[m],\mathrm{r}_1}, ..., \ w_t^{[m],\mathrm{r}_N}, w_t^{[m],\mathbf{o}} \rangle\}_{m=1}^{M}$ to hold temporary weight values for the corresponding robots' and tracked object's temporary sub-particles in the set $\bar{\mathcal{X}}_t$.

In lines $5-10$ of the Algorithm 5.1, $\bar{\mathcal{X}}_t$ stores, after computing, the predicted values of all particles by applying the appropriate robot motion model and the moving object's

Algorithm 5.1 PF-UCLT($\mathcal{X}_{t-1}, \mathbf{u}_t, \mathbf{z}_t, \mathbf{L}_{\text{map}}, \text{r}_k$)

Transmit $\{z_t^{\text{r}_k,\mathbf{o}}$, $u_t^{\text{r}_k}$ and $z_t^{\text{r}_k,l}$ for all $l = 1, ..., L\}$
Receive $\{z_t^{\text{r}_n,\mathbf{o}}$, $u_t^{\text{r}_n}$ and $z_t^{\text{r}_n,l}$ for all $l = 1, ..., L\}$ for all $n = 1, ..., N;\ n \neq k$
$\bar{\mathcal{X}}_t \triangleq \{\langle \bar{\mathbf{x}}_t^{[m]}, \bar{w}_t^{[m]} \rangle\}_{m=1}^{M} \overset{\text{def.}}{=} \{\langle \bar{\mathbf{x}}_t^{[m],\text{r}_1}, ..., \bar{\mathbf{x}}_t^{[m],\text{r}_N}, \bar{\mathbf{x}}_t^{[m],\mathbf{o}}, \bar{w}_t^{[m]} \rangle\}_{m=1}^{M} = \mathbf{NULL}$
$\mathcal{W}_t \triangleq \{\langle w_t^{[m],\text{r}_1}, ..., w_t^{[m],\text{r}_N}, w_t^{[m],\mathbf{o}} \rangle\}_{m=1}^{M} = \mathbf{NULL}$
for $m = 1$ to M **do**
  for $n = 1$ to N **do**
    $\bar{\mathbf{x}}_t^{[m],\text{r}_n} = \textbf{\textit{sample\_robot\_motion\_model}}(\mathbf{x}_{t-1}^{[m],\text{r}_n}, \mathbf{u}_t^{\text{r}_n})$
  end for
  $\bar{\mathbf{x}}_t^{[m],\mathbf{o}} = \textbf{\textit{sample\_object\_motion\_model}}(\mathbf{x}_{t-1}^{[m],\mathbf{o}}, \mathbf{v}_t^{\mathbf{o}})$
end for
for $m = 1$ to M **do**
  for $n = 1$ to N **do**
    $w_t^{[m],\text{r}_n} \propto \prod_{l=1}^{L} p(\mathbf{z}_t^{\text{r}_n,l} \mid \bar{\mathbf{x}}_t^{[m],\text{r}_n}, \mathbf{L}_{\text{map}})$
  end for
end for
for $n = 1$ to N **do**
  $\{\langle \bar{\mathbf{x}}_t^{[m],\text{r}_n}, w_t^{[m],\text{r}_n} \rangle\}_{m=1}^{M} \leftarrow \textbf{\textit{sort\_descend}}\ (\{\langle \bar{\mathbf{x}}_t^{[m],\text{r}_n}, w_t^{[m],\text{r}_n} \rangle\}_{m=1}^{M})$ **w.r.t.** $\{w_t^{[m],\text{r}_n}\}_{m=1}^{M}$
end for
for $m = 1$ to M **do**
  Choose index $m^* \in [1 : M]$ such that the value of $w_t^{[m^*],\mathbf{o}} \propto \prod_{n=1}^{N} p(\mathbf{z}_t^{\text{r}_n,\mathbf{o}} \mid \bar{\mathbf{x}}_t^{[m],\text{r}_n}, \bar{\mathbf{x}}_t^{[m^*],\mathbf{o}})$ is maximum for all possible indices $m^* \in [1 : M]$.
  $\bar{\mathbf{x}}_t^{[m],\mathbf{o}} = \bar{\mathbf{x}}_t^{[m^*],\mathbf{o}}$
  $\bar{w}_t^{[m]} = w_t^{[m^*],\mathbf{o}} \prod_{n=1}^{N} w_t^{[m],\text{r}_n}$
end for
normalize $\{\bar{w}_t^{[m]}\}_{m=1}^{M}$
$\mathcal{X}_t = \textbf{\textit{resample}}(\bar{\mathcal{X}}_t)$
return $\mathcal{X}_t$

motion model on the input particle set $\mathcal{X}_{t-1}$. This is done by separately incorporating the individual robot's odometry measurements and the object's velocity input (the control measurements) on the robot sub-particles $\mathbf{x}_{t-1}^{[m],\text{r}_1}, .., \mathbf{x}_{t-1}^{[m],\text{r}_N}$ and the object sub-particle $\mathbf{x}_{t-1}^{[m],\mathbf{o}}$, respectively. The separate prediction of the sub-particles is as per (5.15) and ***Property 1***,

as stated previously.

Lines $11-23$ present the core of the Algorithm 5.1. The fusion of the correlated observation measurements is done here in an efficient way such that the need of an exponentially high number of particles is avoided. In a PF, the weight generation does not modify the predicted particle set (the predicted state hypothesis of the particles remain unchanged). It only associates a weight to each predicted particle, which is used by the resampling step of the PF. The weighting mechanism presented in the lines $11-23$ also does not modify the predicted particle set. It only associates weights to the predicted particles. To this effect, a rearranging technique of the sub-particles is used which allows the required number of particles M to be much lower than what would be required otherwise.

Lines $11-15$ perform the following: For each robot r_n, its predicted sub-particle $\bar{\mathbf{x}}_t^{[m],\mathrm{r}_n}$ is separately assigned a weight $w_t^{[m],\mathrm{r}_n} \propto \prod_{l=1}^{L} p(z_t^{\mathrm{r}_n,l} \mid \bar{\mathbf{x}}_t^{[m],\mathrm{r}_n}, \mathbf{L}_{\mathrm{map}})$ corresponding to their static landmarks' observation measurements. The separate weighting of the sub-particles is due to ***Property 3***, as stated previously.

In lines $16-18$, each robot's predicted sub-particle set $\{\bar{\mathbf{x}}_t^{[m],\mathrm{r}_n}\}_{m=1}^{M}$ is sorted in descending order w.r.t. to its temporary weight set $\{w_t^{[m],\mathrm{r}_n}\}_{m=1}^{M}$, generated in lines $11-15$. This means that the highest weighted sub-particle, separately for each robot, is assigned the particle index 1 and so on. It is crucial to understand why this is done. Given that a robot's 'good' predicted sub-particles approximate better the correct posterior of that robot's state than its 'bad' sub-particles, consider the following situation. If we were using the straightforward particle weighting mechanism, i.e, the first expression in (5.16), it would be very likely that a robot's 'good' sub-particle would get coupled with a 'bad' sub-particle from another robot when computing the weight of a particle leading to a lower overall weight of that particle, even though a sub-particle in it was 'good'. It is due to this loss of 'good' sub-particles and eventually the loss of 'good' particles that an exponentially high number of particles are required to not fall into the particle deprivation problem when using the first expression of (5.16) directly. By sorting the individual robot's sub-particles and then clubbing the 'best with the best' robots' sub-particles, we make sure that the particle deprivation problem is solved partially (because we still haven't considered the object observation measurements). Furthermore, recall that any robot r_n's sub-particle set $\{\bar{\mathbf{x}}_t^{[m],\mathrm{r}_n}\}_{m=1}^{M}$ was predicted and weighted independently of all other robots' sub-particle sets which implicitly means that rearranging the robot sub-particle sets does not modify the distribution represented by the predicted particle set $\bar{\mathcal{X}}_t$.

Finally, lines $19-23$ incorporate the robots' observation measurements of the tracked object. Following from ***Property 1***, the prediction of the tracked object's sub-particles

$\{\bar{\mathbf{x}}_t^{[1],\mathbf{o}}, ..., \bar{\mathbf{x}}_t^{[M],\mathbf{o}}\}$ was performed independently of the robots' sub-particles prediction. This enables the freedom to rearrange the tracked object's sub-particles before incorporating the object observation measurements in the particle's weight.

We perform this rearrangement of the object's sub-particles to fully solve the problem of particle deprivation (recall that earlier by rearranging only the robot's sub-particles it was partially solved) as follows. For an m^{th} previously-rearranged set of the robots' predicted sub-particles $\{\bar{\mathbf{x}}_t^{[m],\mathrm{r}_1}, ..., \bar{\mathbf{x}}_t^{[m],\mathrm{r}_N}\}$, obtained after the execution of lines $16-18$, we choose the tracked object's sub-particle[2] $\bar{\mathbf{x}}_t^{[m^*],\mathbf{o}}$ such that $w_t^{[m^*],\mathbf{o}}$ is maximum for all possible indices $m^* \in [1:M]$. Here[3], $w_t^{[m^*],\mathbf{o}}$ is computed as $w_t^{[m^*],\mathbf{o}} \propto \prod_{n=1}^{N} p(z_t^{\mathrm{r}_n,\mathbf{o}} \mid \bar{\mathbf{x}}_t^{[m],\mathrm{r}_n}, \bar{\mathbf{x}}_t^{[m^*],\mathbf{o}})$. At this point, the combination of all the predicted sub-particles (robots' and the tracked object's) for an m^{th} particle is complete. Subsequently in line 22, the weight $\bar{w}_t^{[m]}$ of the m^{th} particle $\bar{\mathbf{x}}_t^{[m]}$ is calculated as $\bar{w}_t^{[m]} = w_t^{[m^*],\mathbf{o}} \prod_{n=1}^{N} w_t^{[m],\mathrm{r}_n}$. Note that this is in accordance with the final expression obtained in (5.16). This is done for every m^{th} set of the robots' predicted sub-particles. The selection of the tracked object's sub-particle, in a way such that the 'good' object sub-particle gets coupled with the 'good' sub-particle set of the robots, ensures that the particle deprivation problem is fully solved. This concludes the weight generation step (and therefore the incorporation of all correlated or uncorrelated observation measurements) of the Algorithm 5.1.

Line 24 performs the particle weight normalization followed by the resampling step in line 25. The resampling of the particle set $\bar{\mathcal{X}}_t$ is performed to obtain $\mathcal{X}_t$, which is eventually returned as the final output of this algorithm. This can be done using standard available methods, e.g, low variance resampling [43].

Space and Time Complexity Analysis of PF-UCLT

In this sub-subsection, worst-case space and time computational complexity analysis of the Algorithm 5.1 is presented and compared with a scenario in which, instead of Algorithm 5.1, the traditional PF method (described at the beginning of this section) would be used. We assume the same notations, as introduced previously, that the total number of robots in the team is N and the total number of tracked objects is O. Note that in the Algorithm 5.1's description, we assumed $O = 1$. Therefore, in the complexity analysis presented here, we need to consider that lines $20-22$ are repeated O times for $O > 1$.

[2]Object sub-particle at the index m^* where $m^* \in [1:M]$

[3]This expression follows from ***Property 4***

Furthermore, this complexity analysis will also assume that M is the number of particles required to obtain a reasonably good accuracy by a PF-based method for a single robot localization where the robot's state consists of a 2D pose and an orientation (or for a single object tracking case where the state is the 3D position of the object)[4].

Space Complexity: In a traditional PF-based method for the online UCLT problem the worst-case space complexity will be in the order of $\mathcal{O}\Big(M^{N+O}\Big)$, since the number of particles required would grow exponentially with the increase in dimensions of the full state being estimated [56] [57] (increase in number of robots or objects). The exact reasoning for this was given earlier in this subsection. However, the weight generation mechanism of Algorithm 5.1 limits the worst-case space complexity to $\mathcal{O}\Big((N+O)M\Big)$, since only M particles, with $(N+O)$ sub-particles in each particle, are required. This feature of the PF-UCLT algorithm makes it scalable to a large number of robots and the tracked objects, where the robots might have low-memory capacities. Nevertheless, it implies more communication among the robots and a higher bandwidth usage.

Time Complexity: Although a traditional PF-based method would require only one iteration over the whole particle set for the weight generation step, given the high number of particles used, the worst-case time complexity (WCTC) for it would be in the order of $\mathcal{O}\Big(M^{N+O}\Big)$, i.e, growing exponentially with the number of robots and objects. Algorithm 5.1 substantially reduces the time complexity. The WCTC of the weight generation step due to the fixed landmarks observation measurements (lines $11-15$ of the algorithm) is in the order of $\mathcal{O}\Big(NM\Big)$. The WCTC of the sorting performed in lines $16-18$ (assuming merge sort algorithm used for sorting) will be in the order of $\mathcal{O}\Big(NM\log M\Big)$. For the weight generation step due to the tracked objects' observation measurements (lines $19-23$ with lines $20-22$ repeated O times for O objects), the WCTC will be in the order of $\mathcal{O}\Big(NM^{O+1}\Big)$. Summing all the individual steps' WCTC and considering only the highest order term, while assuming that $M \gg (N+O)$, the WCTC of the weight generation step in Algorithm 5.1 will be in the order of $\mathcal{O}\Big(NM^{O+1}\Big)$. This is linear in terms of the number of robots N in the team, which would otherwise be exponential in a traditional PF solution for the online UCLT problem. Therefore, scaling the Algorithm 5.1 to a larger number of robots is possible, however, the scalability is still limited in terms of O, the number of

[4]Note that, since the required number of particles depends on the state space dimension [56] [57], a higher dimensional space for a single robot's pose or the object would simply require M to increase exponentially with the dimension.

objects being tracked which exponentially increases the required computation time.

5.4.3 Experiments and Results

5.4.3.1 Testbed and Experimental Scenario

PF-UCLT (Algorithm 5.1) was implemented on the same testbed and dataset as in the case of O-MMLG, described earlier in this chapter in sub-subsection 5.3.3.1 (the dataset containing 4 soccer robots' data, their odometry logs as well as 6 landmarks' data and the orange ball data from their observation logs). Since the O-MMLG-based approach is an offline method, it uses the whole dataset at once as a batch for each iteration of the graph optimization process. However, the PF-UCLT approach is an online method, therefore an iteration of its algorithm uses only the data at the corresponding timestep, causing the whole dataset to be used only once.

Algorithm 5.1 was first implemented on only 2 of the robots' data logs to verify the algorithm's proof of concept and run-time feasibility. The results and analysis for it are presented in sub-subsection 5.4.3.2. Later the implementation was extended to all the 4 robots' data logs to analyze the algorithms scalability. The results for it are presented further in sub-subsection 5.4.3.3.

Note that, unlike the O-MMLG method, the velocity of the tracked object in PF-UCLT is not a part of the estimated state. Here, the ball velocity at every timestep is measured using a separate velocity sensor which performs a linear regression over the tracked ball positions during a fixed number of previous timesteps. Subsequently, this ball velocity measure, along with a zero mean Gaussian acceleration noise, is used in the PF-UCLT prediction step as per the last equation of (5.15).

5.4.3.2 2 Robots Experiment

For its proof of concept, Algorithm 5.1 needs to be implemented in a scenario where at least 2 robots are cooperatively localizing and tracking 1 object. To this effect, we implemented Algorithm 5.1 on the robot named OMNI1 (recall that the algorithm is computationally decentralized) assuming that it acquires its own measurement data of odometry, landmarks and orange ball observation and receives the measurement data for the same from the robot OMNI2. In order to emulate communication between the two robots, at every iteration of the algorithm on OMNI1, we chose the closest timestamped data from OMNI2's data log. The data logs of both robots used in this experiment correspond to ~ 6 minutes of real-time data acquisition (For complete details regarding the data logs, see Appendix A).

	PF-UCLT at OMNI1		
	Mean error (m)	Median error (m)	Variance of error (m^2)
OMNI1	0.106	0.105	0.003
OMNI2	0.161	0.146	0.006
Ball (global position)	0.369	0.282	0.068
Ball (local position in OMNI1's frame)	0.246	0.181	0.047

Table 5.4: PF-UCLT: Statistical estimates of the 2 Robots experiment results.

Table 5.4 and Figures 5.5 – 5.7 present the results of this experiment. The left column plots in Figure 5.7 present the error in the global estimated position of the orange ball. The right column plots in Figure 5.7 correspond to the error in local frame range estimates of the orange ball in OMNI1's frame of reference. Error in global position is computed as the absolute value of the difference in the ball's 3D position estimated by Algorithm 5.1 running on OMNI1 and the corresponding GT estimates for the same. The error in OMNI1's local frame is computed as the absolute value of the difference of i) distance between Algorithm 5.1's estimate of OMNI1's position and the ball's position and ii) distance between the corresponding GT estimates of OMNI1 and the orange ball. The error in local frame is in the terms of range (radial distance) of the ball from the observing robot and not explicitly in the local 2-D coordinates X and Y of the observing robot because the GT system provides only the 2D positions of the robots, not the GT orientation of the robots.

In Figure 5.5, localization confidence values of both OMNI1 and OMNI2 are presented. The localization confidence of a robot is inversely proportional to the spatial variance of the sub-particles associated to it in Algorithm 5.1. It must be noted that, also mentioned in the introduction of this chapter, a unified method for cooperative tracking and localization must be independent of such confidence measures since they are purely heuristic-based measures and can often result in false positives. The localization confidence values presented here are not used in the PF-UCLT algorithm anywhere. These are presented for a quick evaluation of the localization performance achieved by Algorithm 5.1 and must be viewed in conjunction with the robot localization error plots of Figure 5.6. The plots in Figure 5.6 present the errors in each robot's localization estimates and are computed as the absolute value of the difference of i) the robot's 2D position (out of 2D position and orientation estimates) estimated by Algorithm 5.1 running on OMNI1 and ii) corresponding 2D position of the robot obtained by the GT system. The robots often get occluded from

the GT system, hence the localization error values in Figure 5.6 are not available at all timesteps of the experiment. Therefore a heuristic-based confidence value of localization helps, to some extent, better visualize the localization performance by PF-UCLT. In case of the 4 robots experimental results, presented in the next sub-subsection, the localization performance can be visualized even better using the video accompanying this section. All the plots, except the histograms, have a common X-axis scale and limits since they correspond to the same timesteps of Algorithm 5.1.

Table 5.4 presents the statistical estimates of the aforementioned errors. These results show that PF-UCLT (Algorithm 5.1) was able to cooperatively localize OMNI1 and OMNI2 and track the orange ball with good accuracy, comparable to that of the O-MMLG approach. The mean error of localization for both robots is only slightly more (~ 0.03m) than what was achieved using the offline method O-MMLG for the 2 robots experiment in sub-subsection 5.4.3.2. The mean error in the estimated global position of the ball is 0.369m which in comparison with the O-MMLG method is less by 0.06m. The mean error of the estimated local position of the ball in OMNI1's reference frame is 0.246m, much less than its global position mean error, which reflects that the error in the global position of the ball incorporates the observing robots' localization errors. The variance of error in both robots' localization estimates is similar to their O-MMLG counterpart. However, it must be noted that the PF-UCLT method results in a much smoother trajectory of the estimated ball positions than the O-MMLG method. This is evident by the fact that the variance of error in the global positions of the ball is 0.068m^2 when using the PF-UCLT method, while it is 0.314m^2 when using the O-MMLG method.

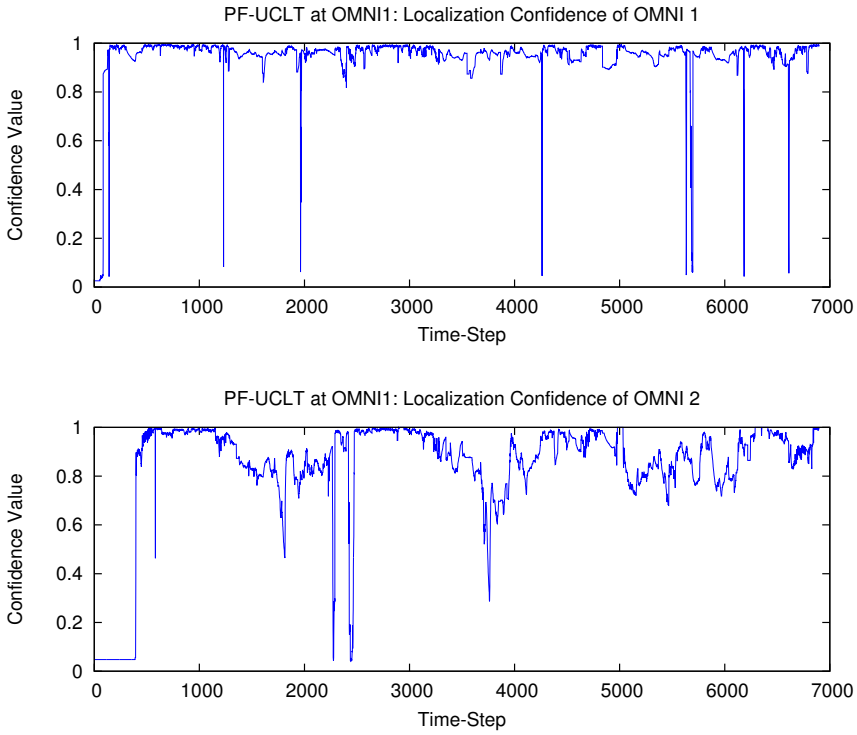


Figure 5.5: PF-UCLT: robots' localization confidence plots for the 2 robots experiment.

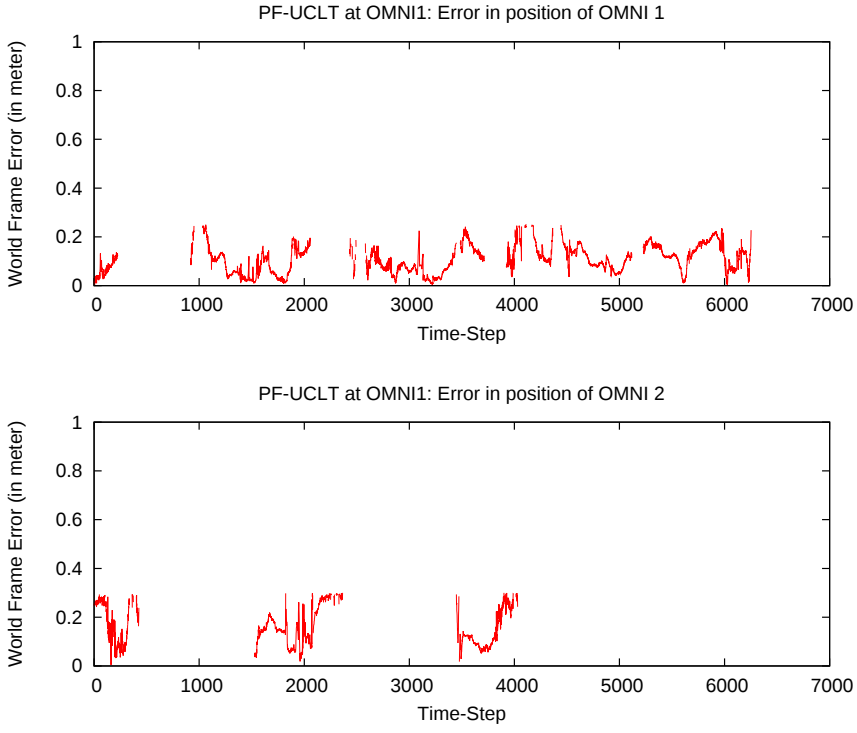


Figure 5.6: PF-UCLT: robot's estimated position error plots for the 2 robots experiment.

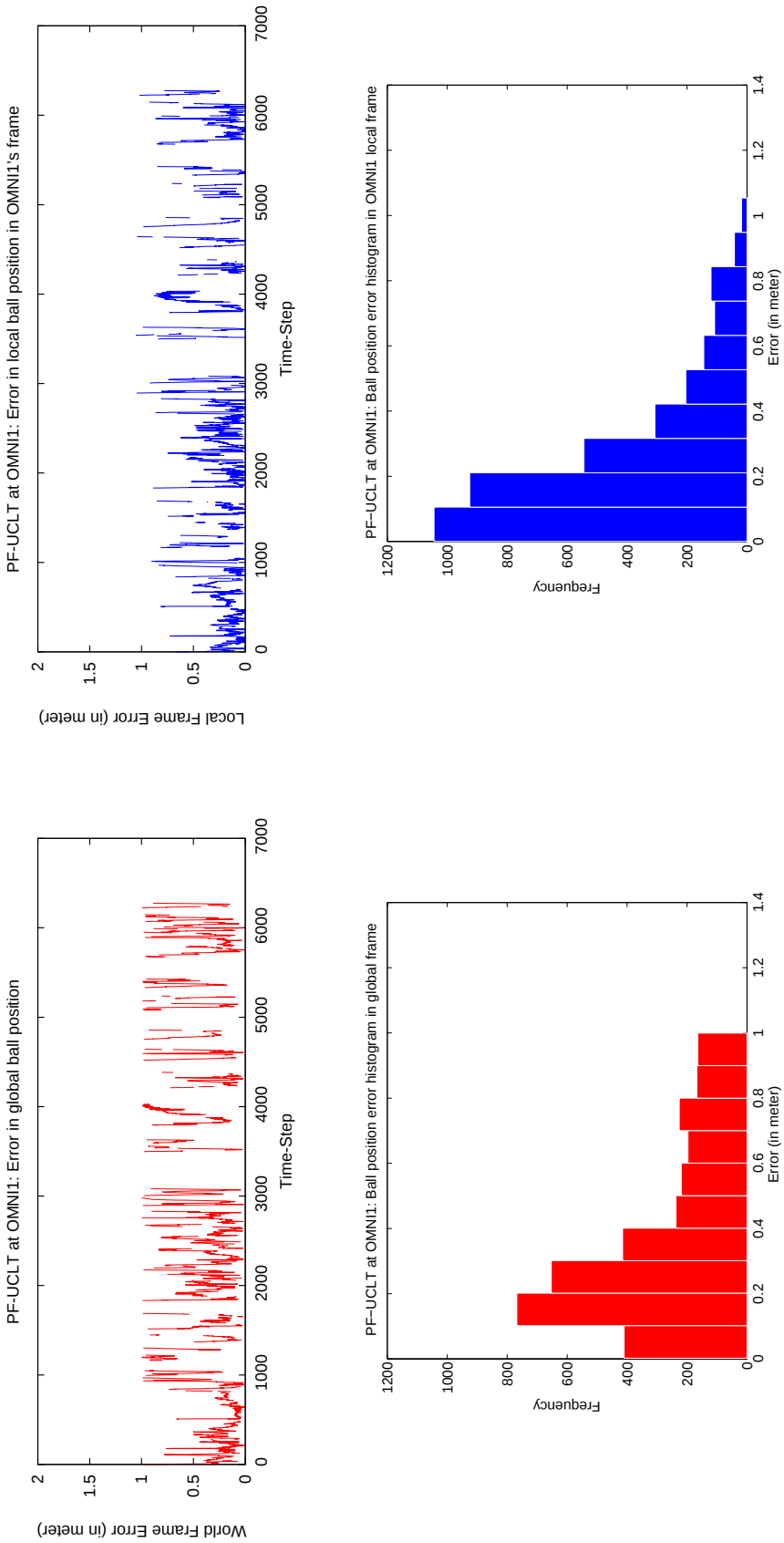


Figure 5.7: PF-UCLT at OMNI1 for the 2 robots experiment: The top left plot presents the tracked ball's global frame position estimate errors. The top right plot presents the tracked ball's local frame range (radial distance to the ball from the observing robot) estimate errors in OMNI1's frame. The corresponding bottom row plots present the histograms of these errors. All errors are computed as explained in the sub-subsection 5.4.3.2

5.4.3.3 4 Robots Experiment

	PF-UCLT at OMNI1		
	Mean error (m)	Median error (m)	Variance of error (m^2)
OMNI1	0.103	0.101	0.003
OMNI2	0.159	0.157	0.006
OMNI3	0.126	0.120	0.003
OMNI4	0.339	0.239	0.069
Ball (global position)	0.338	0.279	0.050
Ball (local position in OMNI1's frame)	0.225	0.169	0.042

Table 5.5: PF-UCLT: Statistical estimates of the 4 Robots experiment results.

In this experiment we extend the PF-UCLT's implementation to the all the 4 robots' data in the dataset. The aim is to analyze the scalability of Algorithm 5.1 in terms of required computation time and its ability to improve the estimates of each entity (robots' pose and tracked object's position estimates) with the increase in number of robots. The robots are henceforth mentioned as OMNI1 – OMNI4. Algorithm 5.1 is implemented on OMNI1 and in addition to its own measurement data, it receives the same from OMNI2, OMNI3 and OMNI4. The communication among the robots is emulated in the same way as explained for the 2 robots experiment in the previous sub-subsection. Table 5.5 and Figures 5.8 – 5.10 present the results of this experiment. The plots in these figures represent the same quantities as that of the 2 robots experiments, explained in the previous sub-subsection. The errors in the local position of the ball is estimated in OMNI1's reference frame. Computation time comparisons for the 4 robot and 2 robot experiments are done in subsequent sub-subsection 5.4.3.4.

In comparison with the 4 robots experiment made using the O-MMLG approach (sub-subsection 5.3.3.3), we observe that Algorithm 5.1 results in slightly more erroneous localization of all the robots, although it is still acceptably good, given that PF-UCLT is an online approach whereas O-MMLG performs the estimations offline. However, the ball tracking accuracy obtained by Algorithm 5.1 is improved by 0.06m when compared to the O-MMLG approach.

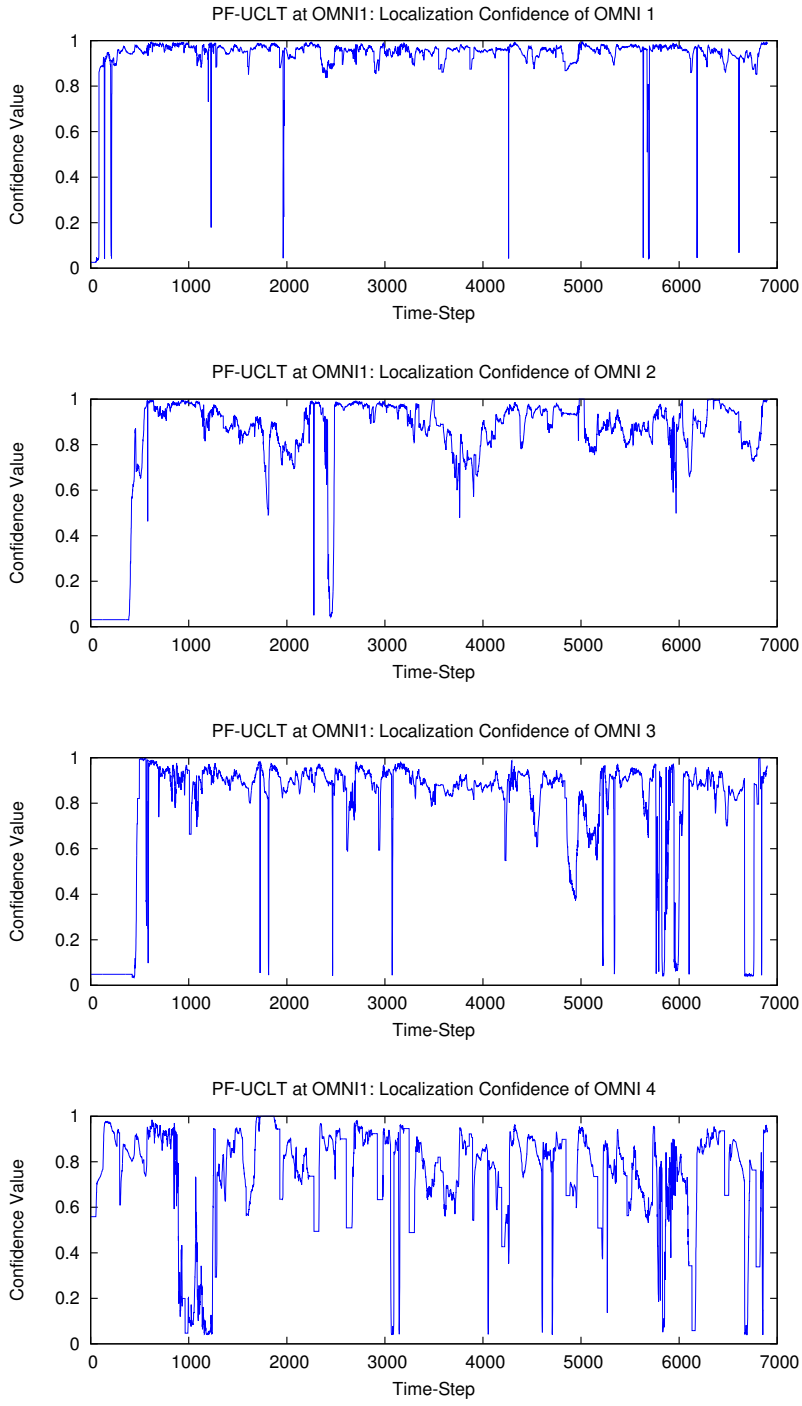


Figure 5.8: PF-UCLT: robots' localization confidence plots for the 4 robots experiment.

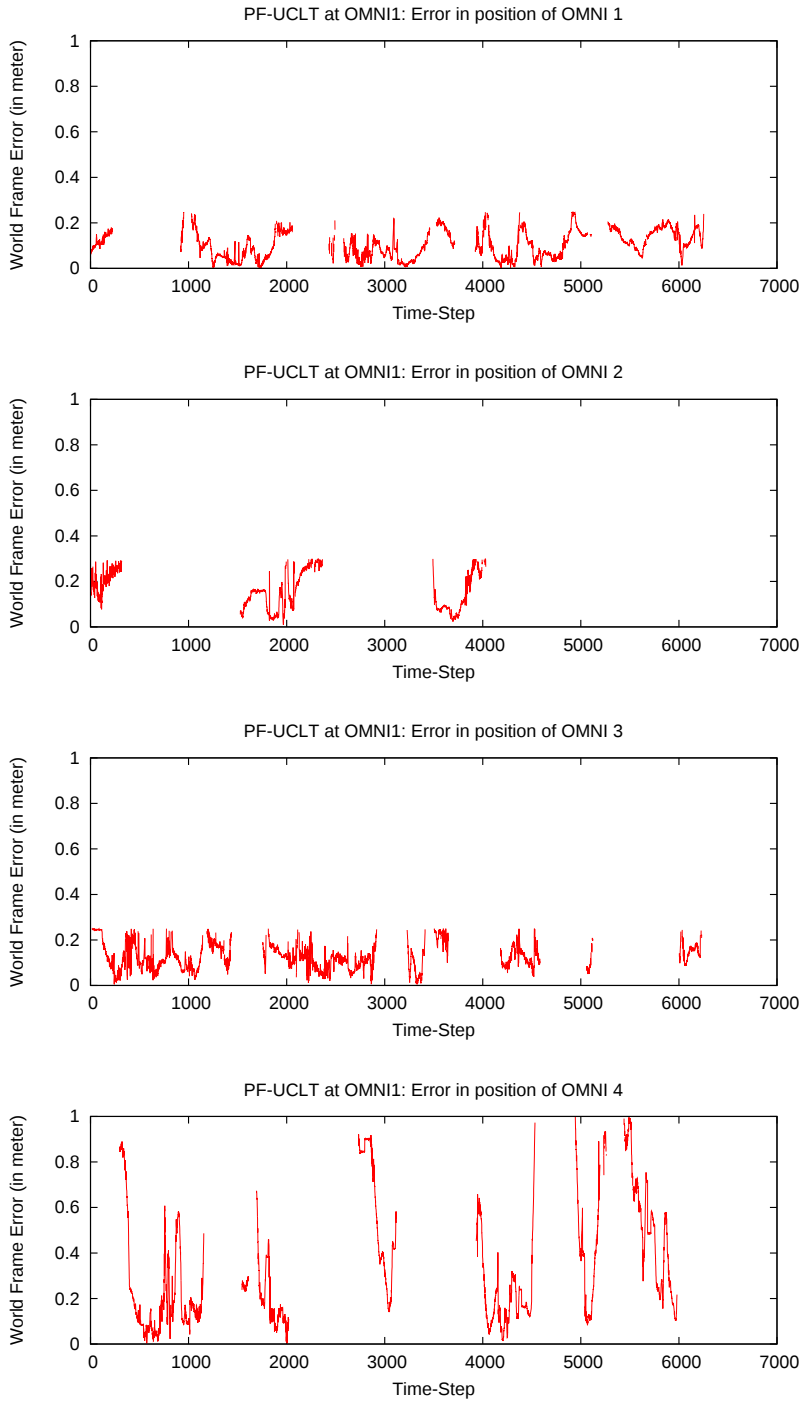


Figure 5.9: PF-UCLT: robots' estimated position error plots for the 4 robots experiment.

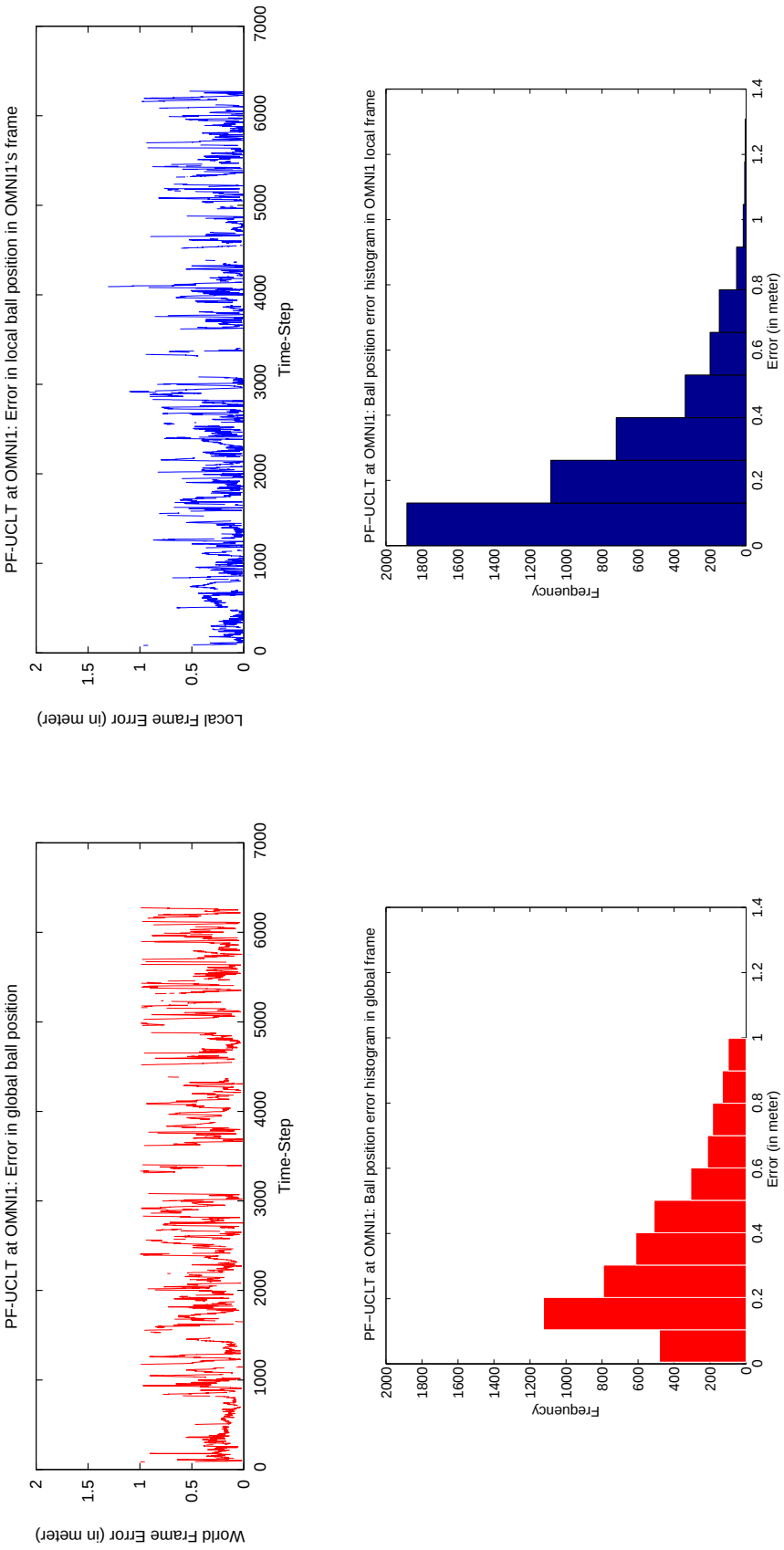


Figure 5.10: PF-UCLT at OMNI1 for the 4 robots experiment: The top left plot presents the tracked ball's global frame position estimate errors. The top right plot presents the tracked ball's local frame range (radial distance to the ball from the observing robot) estimate errors in OMNI1's frame. The corresponding bottom row plots present the histograms of these errors. All errors are computed as explained in the sub-subsection 5.4.3.2

OMNI4 has a much higher localization error compared to the rest of the robots. This can be observed in the error plots for it in Figure 5.9 and the localization confidence plots in Figure 5.8. This is due to the highly noisy measurement data (both odometry and visual observatrions) acquired by it during data acquisition. The O-MMLG approach was able to localize it much better (where OMNI4's mean error = 0.267m) than PF-UCLT (where OMNI4's mean error = 0.339m). This is because the O-MMLG approach is an offline optimizer, where the whole dataset is processed at once and used in every optimization iteration. The iterative re-linearization and batch processing helps mitigate the effect of noisy measurements. This is not possible in a PF-based approach. If noisy measurements exist for a significant period of time, the PF-based tracker estimates poorly during that period. However, it recovers to better estimates when less noisy measurements arrive. Unlike the offline approach of O-MMLG, PF-UCLT is not designed to update the previous trajectory at a given timestep, hence the mean error of localization tends to be higher in its case.

The video[5] accompanying this section displays the 4 robots experiment's results. The overlaid attributes of the video show PF-UCLT and GT estimates in a similar manner as that of the O-MMLG experiment video, explained previously in section 5.3.3. The additional attributes in this video are: i) on top of each robot the word 'OK' or 'LOST' is overlaid depending on whether the associated robot's localization confidence is above a threshold (0.3) or not (see Figure 5.8 for localization confidence values), ii) sub-particles corresponding to each robot and the ball are overlaid in a color corresponding to the robot hat's color and the ball's color, respectively. The overlaid orange circle, representing the ball's estimated position by PF-UCLT, can be observed to perform a much smoother trajectory as compared to the O-MMLG experiment video. An interesting observation is that with OMNI4, the robot with poor localization. Whenever the ball is out of the vision range ($\sim$ 3m) of OMNI4, its localization tends to degrade, however, the moment the ball is observed by OMNI4 again (while being simultaneously observed by any other robot), OMNI4's localization is recovered. This visually demonstrates, one of the key features of the PF-UCLT algorithm, that a cooperatively tracked object enables the poorly localized observing robots to recover from localization failures without actually losing the quality of the cooperatively tracked object's estimate.

[5]Video link of the PF-UCLT 4 robots experiment.
http://users.isr.ist.utl.pt/~aahmad/PhDThesisVideos/Chap_5_PF-UCLT/PF_UCLT.mpg

5.4.3.4 Computation Time Comparison

Both the 2 robots and 4 robots experiment were performed on a machine (Quad Core Intel(R) Core(TM) i5 CPU 750 @ 2.67GHz, 8GB RAM), same as the one used for the O-MMLG approach. As expected according to the theoretical analysis of PF-UCLT's computational complexity, presented earlier in this section, the 4 robots experiment took nearly twice the time (0.05s) per iteration of the PF-UCLT algorithm, compared to the 2 robots experiment in which it took 0.027s to perform an iteration. One iteration refers to the execution of the full Algorithm 5.1 at one timestep. This experimentally establishes that the computational time complexity of Algorithm 5.1 grows only linearly w.r.t. the number of robots in the team.

	Average Time (in seconds) (for 1 iteration of PF-UCLT at OMNI1)
2 Robots Experiment	0.027
4 Robots Experiment	0.050

Table 5.6: Computation time of the PF-UCLT (Algorithm 5.1) method.

5.5 Summary

In this chapter we presented two novel approaches for unified cooperative localization of a team of robots tracking a moving object cooperatively while using it in turn to improve their own localization.

In the first approach (Section 5.3), which is an offline method, we did this by posing it as a multi-robot moving landmark graph optimization problem (O-MMLG), where the moving landmark is the cooperatively tracked object. A mathematical formulation of this problem was presented which verifies its validity and its applicability in the g^{2}o framework for graph optimization. The results of this approach applied in a robot soccer scenario and its comparison with an EKF-based approach highlights its increased accuracy as well as its scalability to a higher number of robots without losing on the accuracy and the required computation time. We also displayed that in the situations of highly noisy measurements and observations, where the EKF-based method performs poorly, the O-MMLG approach was able to produce reasonably good estimates.

In the second approach (Section 5.4), which is an online method, we introduced the PF-based unified cooperative localization and tracking (PF-UCLT) algorithm. This in-

cluded the mathematical formulation of the online UCLT problem, a recursive Bayesian filter solution and description of a standard PF-based method to solve the problem. Subsequently, we presented the details of the PF-UCLT algorithm, describing its novelty, which lies in its update step (particles' weight generation step). We demonstrated that PF-UCLT algorithm reduces the exponentially growing computational and space complexity of a standard PF-based method to linear w.r.t the number of robots in the team. The approach was implemented on the exact same testbed and dataset as the one used in the O-MMLG's case. Results show that the PF-UCLT's accuracy of the full state estimation is reasonably good. However, its accuracy is slightly worse than the O-MMLG approach's accuracy. As pointed out before, since the O-MMLG performs the estimation of the full state's complete trajectory by iteratively optimizing over the whole measurement dataset in every iteration, its higher accuracy over the PF-UCLT algorithm (online approach) is explicable. We analyzed the runtime computational space and time complexity for the PF-UCLT method, both theoretically and experimentally, and concluded that it is well-suited for realtime applications. We also presented a one to one comparison of both the approaches while highlighting their capabilities and execution speed.

In Chapter 6, we present a more detailed experimental comparison of both the unified methods presented here by applying them in various experimental failure situations, e.g, communication failure between robots and sensor breakdown.

5.6 Related Publications

The work related to O-MMLG method for offline unified cooperative localization and object tracking (UCLT), presented in Section 5.3, was accepted as a full length-article [8] to be published in the proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013), Karlsruhe, Germany (submitted: September 2012, accepted: January, 2013).

Chapter 6

Case Study 1: Comparative Analysis of the Cooperative Perception Algorithms

In this case study, we present a comparative analysis of the cooperative perception (CP) algorithms developed in this thesis. The feature comparison, in section 6.1, presents a chart summarizing the requirements, advantages/disadvantages, failure situations, etc., of the CP algorithms. The comparison highlights which particular CP algorithm is more suitable for implementation in a certain scenario.

The quantitative analysis, presented in section 6.2, explores the robustness of the unified CP algorithms, PF-UCLT and O-MMLG, developed in Chapter 5. This is done through a series of experiments, where we emulate failure situations, eg., communication loss and vision system's hardware failure. The testbed and dataset used for these experiments are described in Appendix A.

6.1 Feature Comparison

Inferring from the algorithm descriptions and the experimental results presented in the preceding chapters, Table 6.1 presents a comprehensive feature list of the following CP algorithms/methods:

- A PF-based algorithm for multi-robot cooperative object tracking (PF-MCOT), described in Chapter 3.
- A PF-based algorithm for multi-robot cooperative robot localization (PF-MCRL),

described in Chapter 4.

- Multi-robot moving landmark graph optimization (O-MMLG) method for multi-robot unified cooperative localization and object tracking, described in Chapter 5, Section 5.3.
- A PF-based algorithm for multi-robot unified cooperative localization and object tracking (PF-UCLT), described in Chapter 5, Section 5.4.

Feature	PF-MCOT	PF-MCRL	O-MMLG	PF-UCLT
Necessity of a separate robot localization method	Yes	No	No	No
Necessity of a separate object tracking method	No	Yes	No	No
Integrated robot localization and object tracking	No	No	Yes	Yes
Heuristics-based robot localization confidence necessary	Yes	Yes	No	No
Heuristics-based object tracking confidence necessary	Yes	Yes	No	No
Recursive error propagation	No	No	No	No
Robust to communication failures	Yes	Yes	Yes	Yes
Robust to sensor failures of teammates	Yes	Yes	Yes	Yes
Fully decentralized	Yes	Yes	No	Yes
Scalable to a large number of robots	Yes	Yes	Yes	Yes
Suitable for online implementation	Yes	Yes	No	Yes
Full trajectory estimation	No	No	Yes	No

Table 6.1: Feature Comparison of the CP algorithms

6.2 Quantitative Analysis

We performed unified cooperative multi-robot localization and object tracking experiments using the O-MMLG and PF-UCLT methods on the "4_robots_dataset" (one of the datasets

described in Appendix A) for localizing the robots and to track the orange ball. All 4 robots', OMNI1–OMNI4's, odometry log and 6 landmarks observation logs were used along with their observation logs for the orange ball. The corresponding ground truth (GT) logs were used to quantify errors in the robots' 2D position estimates and the ball's 3D position estimates. The dataset used here corresponds to ~ 6 minutes of realtime data acquisition.

In order to do a robustness check and analyze the algorithm's behavior in various failure scenarios, a set of 12 experiments (3 experimental situations in 4 different scenarios[1]) were designed and both the O-MMLG method and the PF-UCLT algorithm were implemented in each experiment. Since the PF-UCLT method is decentralized, results of its implementation on the robot OMNI1 is analyzed. The scenarios are as follows:

- Permanent communication failure (results presented in subsection 6.2.1)
- Temporary communication failure (results presented in subsection 6.2.2)
- Permanent vision failure (results presented in subsection 6.2.3)
- Temporary vision failure (results presented in subsection 6.2.4)

Each of the above scenarios consists of 3 different experimental situations[1], which correspond to different number of failed robots in that scenario. To emulate the communication or the vision failure, the data logs (both odometry and camera observation logs in case of communication failure and only camera observation logs in case of vision failure) corresponding to specific timestamps were skipped from the algorithm's execution process. In the subsequent sub-sections, we present the results of all the experiments, grouped according to the failure scenarios. Along with a table of results for each experiment, the box-style plots in Figures 6.1 – 6.8 help visualize the evolution of estimation errors with the increasing number of failed robots. In the box-style plots, for each experiment, the center of the rectangular box is positioned at the X-coordinate corresponding to the experiment ($X = 1$ for OMNI2's failure experiment situation, $X = 2$ for OMNI2 and OMNI3's failure experiment situation and $X = 3$ for OMNI2, OMNI3 and OMNI4's failure experiment situation) and the Y-coordinate corresponding to the mean error of estimation (mean error of a robot's or the ball's position estimate) in that particular experiment. A horizontal line inside the box denotes the median of the error, while the lower and higher edge of the

[1]A scenario corresponds to the kind of failure, e.g, communication failure between robots or a particular robot losing its vision sensor. An experimental situation corresponds to the number of robots failing in a particular scenario.

box correspond to the 25^{th} and 75^{th} percentile error of estimation, respectively. The end points of the 'whiskers' projecting out from either sides of the box, depict the highest and lowest errors of the experiment. These statistics for each experiment are of the O-MMLG or PF-UCLT's estimates (and errors calculated by comparing these estimates with the ground truth) on the dataset corresponding to 6 minutes of realtime data acquisition, as mentioned previously in this section.

Prior to the analysis of results, it is worth mentioning that in the dataset used in all the experiments here, the individual robot's odometry noise and observation measurement noise were much higher in the case of OMNI4 where as quite low in the case of OMNI1. This was mainly due to to the high slippage in OMNI4's motors causing a poor odometry reading from it and poor color segmentation calibration for its vision system causing noisy observation measurements.

6.2.1 Permanent Communication Failure

This sub-section details the following 3 experimental situations (in increasing order of the number of failed robots) under the scenario of a permanent communication failure.

1 Failed Robot: OMNI2 loses communication with OMNI1 from $\sim 80^{\text{th}}$s up till the end.

2 Failed Robots: OMNI2 loses communication with OMNI1 from $\sim 80^{\text{th}}$s up till the end and OMNI3 loses communication with OMNI1 from $\sim 85^{\text{th}}$s up till the end.

3 Failed Robots: OMNI2 loses communication with OMNI1 from $\sim 80^{\text{th}}$s up till the end, OMNI3 loses communication with OMNI1 from $\sim 85^{\text{th}}$s up till the end and OMNI4 loses communication with OMNI1 from $\sim 92^{\text{nd}}$s up till the end.

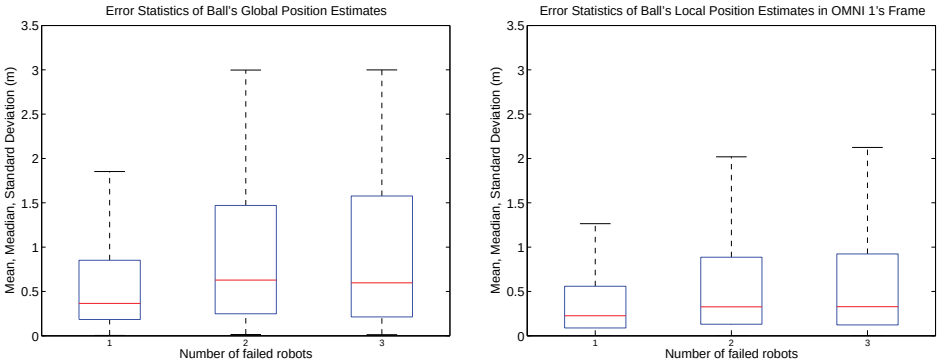
The end of all the experiments here and in the subsequent sub-sections is at $\sim$ 360s. Table 6.2 and the plots in Figures 6.1 and 6.2 present the results of the PF-UCLT algorithm and the O-MMLG method applied in this scenario.

Primarily, we observe that for all the three experiments here, the errors for all the robots' and the ball's position estimates are lower when the O-MMLG method is applied in comparison with the PF-UCLT algorithm's implementation. As previously mentioned in Chapter 5, this is due the fact that the O-MMLG method, being an online full trajectory estimator, iterates over the whole dataset multiple times to achieve the optimal configuration of nodes in the pose graph. On the other hand, PF-UCLT is an online estimator which process a measurement data only once. Therefore, while gaining on computational speed

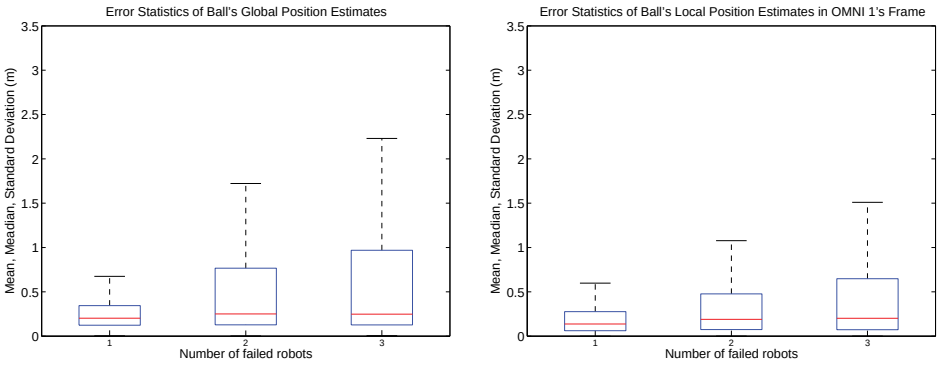
and realtime execution ability, PF-UCLT compromises on accuracy to some extent. Nevertheless, the accuracy of both methods are reasonably good even after the communication loss.

In both the unified CP approaches, PF-UCLT and O-MMLG, we observe from the plots in Figure 6.2 that the position estimation accuracy of the 'disconnected' robots (those robots which lose communication with OMNI1) degrades while at the same time the position accuracy of the 'connected' robots (those robots whose communication with OMNI1 has not failed) is maintained. It is notable that OMNI1, on which the algorithm runs in the case of PF-UCLT or whose self measurement/observation data is never lost in case of the O-MMLG approach, is able to maintain its position estimate accuracy when one or more teammates' communication fails. This particularly highlights the robustness of both the unified CP approaches in the case of permanent communication loss. It is interesting to observe that when OMNI4 is 'disconnected' (loses communication), OMNI1 slightly improves its own position estimate accuracy (mean error reduces from 0.100m to 0.093m in the case of PF-UCLT and from 0.64m to 0.62m in the case of O-MMLG approach) while OMNI4's position accuracy reduces drastically (mean error increases from 1.40m to 2.76m in the case of PF-UCLT and from 0.187m to 2.153m in the case of O-MMLG approach). The main reason behind it is that OMNI4's measurements are corrupted with high noise. Such a phenomena highlights a beneficial feature of our unified CP approaches. The position estimates of the robots, with highly noisy measurements, improve quite significantly in the presence of robots with less noisy measurements, while at the same time the latter robots' position accuracy is only slightly compromised.

The accuracy of the ball's position estimate degrades with the number of robots losing communication. This is due to the fact that once a robot is 'disconnected' there are less ball observation measurements, (recall that in this dataset a robot observes a ball only up to $\sim$ 3m distance from it) hence for significant periods of time, the ball might simply not be visible, causing an increase in the overall mean error of its position estimation. The changes in the median of the ball's position errors show another interesting feature of our methods. In both the CP approaches, the median of the ball's position errors increases significantly when OMNI3 and OMNI2 are 'disconnected' (median error increases from 0.365m to 0.628m in the case of PF-UCLT and from 0.202m to 0.250m in the case of O-MMLG approach) and only slightly decreases when OMNI4 is also 'disconnected' along with OMNI2 and OMNI3 (median error decreases from 0.628 to 0.587 in the case of PF-UCLT and from 0.250m to 0.248m in the case of O-MMLG approach). This shows that while the less noisy observations of OMNI2 and OMNI3 (their noise was comparable to that of OMNI1's) were improving the overall ball's estimate, the more noisy ball ob-



PF-UCLT



O-MMLG

Figure 6.1: The box plots in this figure present the statistical estimates of the ball's global (and local in OMNI1's frame) position estimation errors for the PF-UCLT and the O-MMLG experiments performed in the scenario of permanent communication failure. The X-axis represents the three experimental situations, described in the sub-section 6.2.1, of this scenario.

servation measurements of OMNI4 were only slightly degrading the overall ball's position estimate. Therefore establishing that our unified CP approaches are robust to more noisy measurements and effectively handle them as outliers.

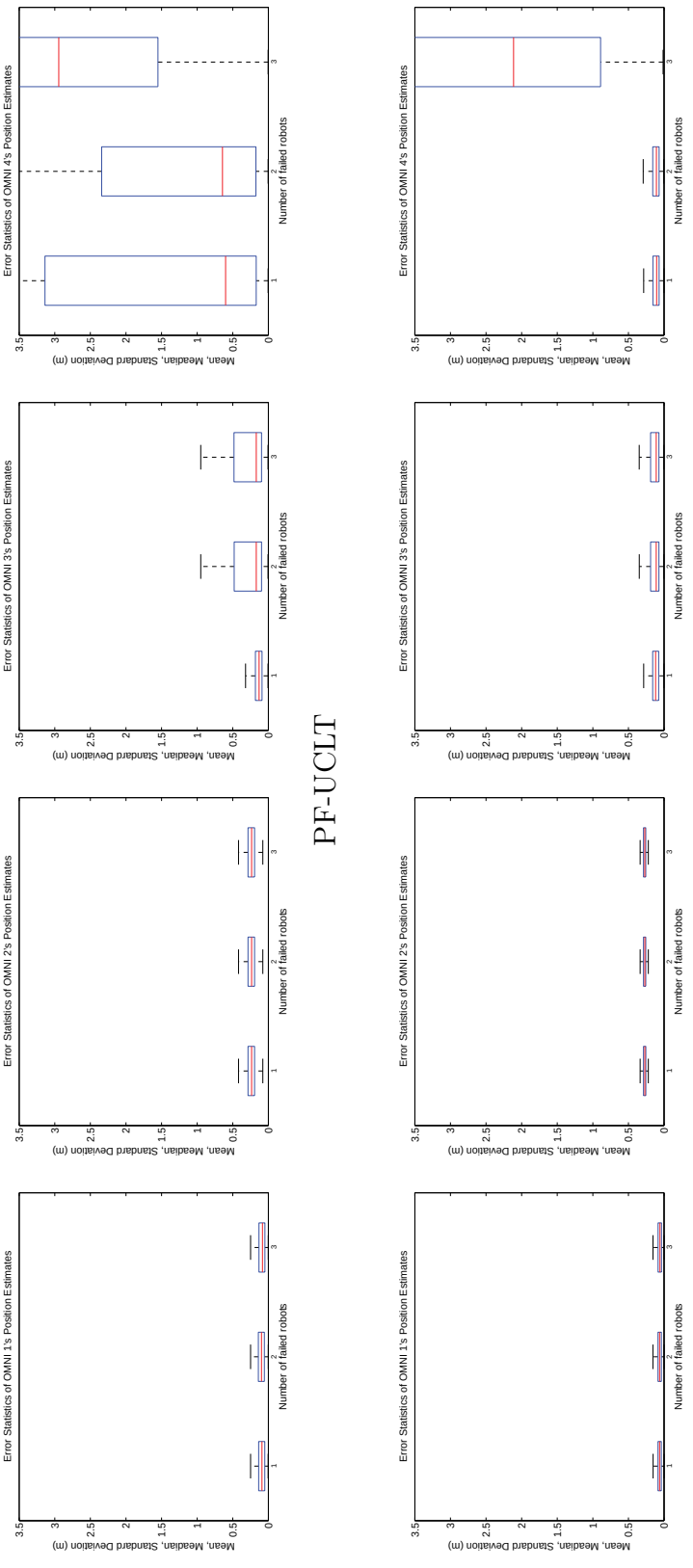


Figure 6.2: The box plots in this figure present the statistical estimates of all the robots' position estimation errors for the PF-UCLT and the O-MMLG experiments performed in the scenario of permanent communication failure. The X-axis represents the three experimental situations, described in the sub-section 6.2.1, of this scenario.

		PF-UCLT			O-MMLG		
Entity	Failed Robot(s)	Mean (m)	Median (m)	Std.dev. (m)	Mean (m)	Median (m)	Std.dev. (m)
OMNI1	OMNI2	0.096	0.090	0.056	0.064	0.062	0.035
	OMNI2,3	0.100	0.097	0.055	0.064	0.063	0.034
	OMNI2,3,4	0.093	0.083	0.054	0.062	0.057	0.035
OMNI2	OMNI2	0.243	0.236	0.075	0.273	0.269	0.023
	OMNI2,3	0.243	0.236	0.075	0.274	0.270	0.023
	OMNI2,3,4	0.243	0.236	0.075	0.274	0.270	0.023
OMNI3	OMNI2	0.152	0.130	0.102	0.123	0.119	0.068
	OMNI2,3	0.295	0.170	0.267	0.138	0.110	0.111
	OMNI2,3,4	0.294	0.170	0.266	0.138	0.110	0.111
OMNI4	OMNI2	1.547	0.601	1.616	0.201	0.104	0.434
	OMNI2,3	1.400	0.645	1.513	0.187	0.107	0.393
	OMNI2,3,4	2.765	2.944	1.715	2.153	2.114	1.536
Ball Global Position	OMNI2	0.657	0.365	0.697	0.334	0.202	0.413
	OMNI2,3	0.904	0.628	0.778	0.607	0.250	0.744
	OMNI2,3,4	0.959	0.597	0.870	0.662	0.248	0.788
Ball Local Position	OMNI2	0.429	0.226	0.531	0.244	0.137	0.344
	OMNI2,3	0.583	0.326	0.608	0.403	0.190	0.537
	OMNI2,3,4	0.598	0.328	0.647	0.465	0.202	0.590

Table 6.2: Error statistics in the situation of permanent communication failure.

6.2.2 Temporary Communication Failure

This sub-section describes the following 3 experimental situations (in increasing order of the number of failed robots) under the scenario of a temporary communication failure.

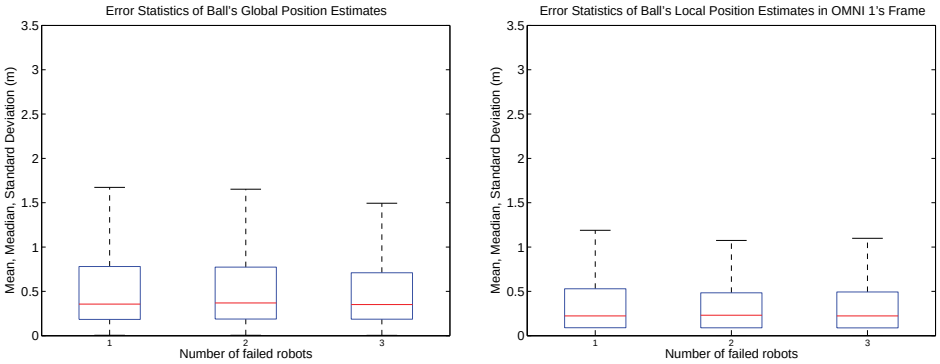
1 Failed Robot: OMNI2 loses communication with OMNI1 from $\sim 80^{\text{th}}$s up to $\sim 120^{\text{th}}$s.

2 Failed Robots: OMNI2 loses communication with OMNI1 from $\sim 80^{\text{th}}$s up to $\sim 112^{\text{nd}}$s and OMNI3 loses communication with OMNI1 from $\sim 85^{\text{th}}$s up to $\sim 120^{\text{th}}$s.

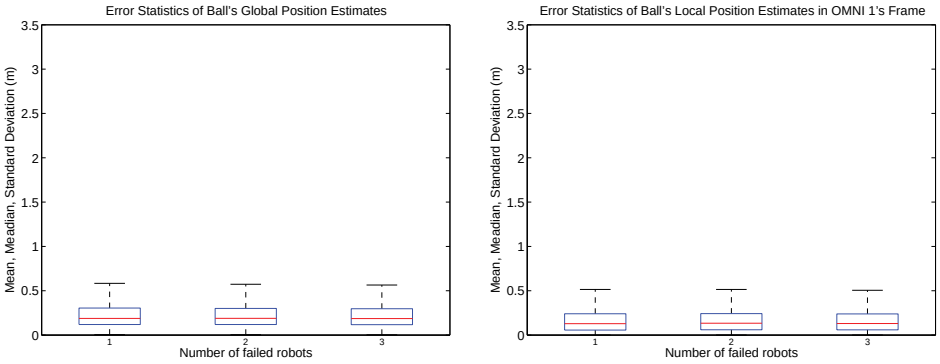
3 Failed Robots: OMNI2 loses communication with OMNI1 from $\sim 80^{\text{th}}$s up to $\sim 112^{\text{nd}}$s, OMNI3 loses communication with OMNI1 from $\sim 85^{\text{th}}$s up to $\sim 120^{\text{th}}$s and OMNI4 loses communication with OMNI1 from $\sim 92^{\text{nd}}$s up to $\sim 128^{\text{th}}$s.

Table 6.3 and the plots in Figures 6.3 and 6.4 present the results of the PF-UCLT algorithm and the O-MMLG method applied in this scenario.

Similar to the previous scenario, the mean errors in the position estimates of all the robots and the ball are comparatively lesser in the case of O-MMLG method than the PF-UCLT, although both methods were able to effectively localize the robots and track the ball. An important inference from the experimental results in this scenario is the stability of errors. With the increase in the number of failed robots, the mean/median errors of all the robots and the ball position estimates change only slightly $(0-10\%)$. This means that both the unified CP methods, O-MMLG and PF-UCLT, are robust to temporary communication failures and able to recover the temporarily 'disconnected' robot's localization estimates, once they get 're-connected' (regained communication).



PF-UCLT



O-MMLG

Figure 6.3: The box plots in this figure present the statistical estimates of the ball's global (and local in OMNI1's frame) position estimation errors for the PF-UCLT and the O-MMLG experiments performed in the scenario of temporary communication failure. The X-axis represents the three experimental situations, described in the sub-section 6.2.2, of this scenario.

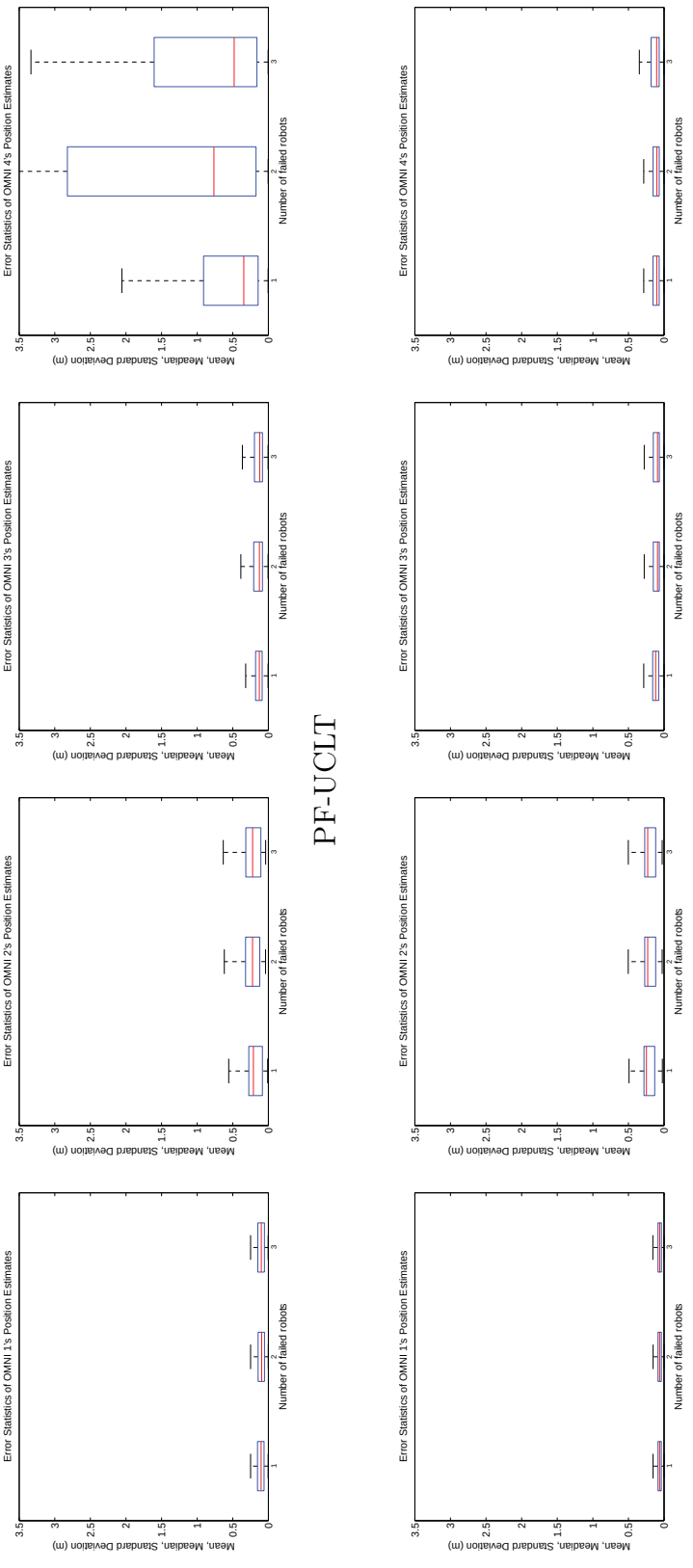


Figure 6.4: The box plots in this figure present the statistical estimates of all the robots' position estimation errors for the PF-UCLT and the O-MMLG experiments performed in the scenario of temporary communication failure. The X-axis represents the three experimental situations, described in the sub-section 6.2.2, of this scenario.

		PF-UCLT			O-MMLG		
Entity	Failed Robot(s)	Mean (m)	Median (m)	Std.dev. (m)	Mean (m)	Median (m)	Std.dev. (m)
OMNI1	OMNI2	0.107	0.102	0.056	0.064	0.061	0.035
	OMNI2,3	0.099	0.094	0.055	0.064	0.061	0.035
	OMNI2,3,4	0.104	0.100	0.058	0.063	0.061	0.035
OMNI2	OMNI2	0.208	0.210	0.145	0.215	0.248	0.104
	OMNI2,3	0.254	0.222	0.169	0.233	0.229	0.169
	OMNI2,3,4	0.253	0.222	0.177	0.232	0.229	0.169
OMNI3	OMNI2	0.149	0.127	0.105	0.123	0.118	0.068
	OMNI2,3	0.181	0.126	0.171	0.117	0.092	0.095
	OMNI2,3,4	0.174	0.122	0.164	0.117	0.092	0.095
OMNI4	OMNI2	0.718	0.345	0.852	0.184	0.102	0.376
	OMNI2,3	1.472	0.765	1.502	0.183	0.100	0.374
	OMNI2,3,4	0.927	0.480	0.953	0.341	0.103	0.625
Ball Global Position	OMNI2	0.621	0.356	0.649	0.282	0.188	0.333
	OMNI2,3	0.621	0.369	0.650	0.298	0.188	0.391
	OMNI2,3,4	0.594	0.351	0.632	0.286	0.186	0.359
Ball Local Position	OMNI2	0.429	0.224	0.521	0.202	0.129	0.272
	OMNI2,3	0.384	0.231	0.454	0.205	0.133	0.273
	OMNI2,3,4	0.410	0.223	0.506	0.198	0.131	0.260

Table 6.3: Error statistics in the situation of temporary communication failure.

6.2.3 Permanent Vision Failure

This sub-section describes the following 3 experimental situations (in increasing order of the number of failed robots) under the scenario of a permanent vision failure.

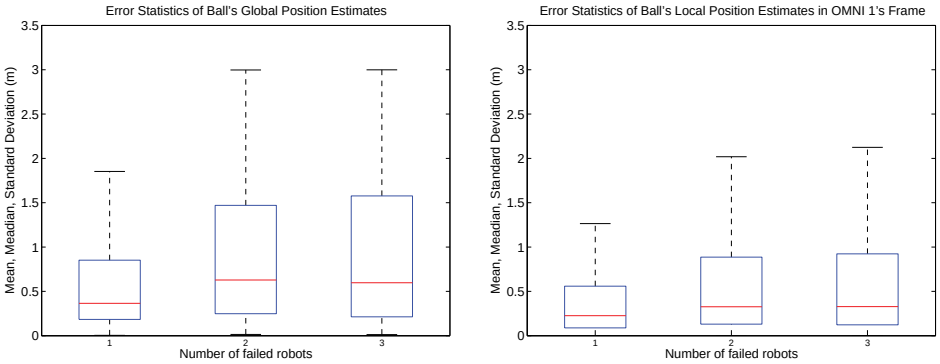
1 Failed Robot: OMNI2 loses camera vision from $\sim 80^{\text{th}}$s up till the end.

2 Failed Robots: OMNI2 loses camera vision from $\sim 80^{\text{th}}$s up till the end and OMNI3 loses camera vision from $\sim 85^{\text{th}}$s up till the end.

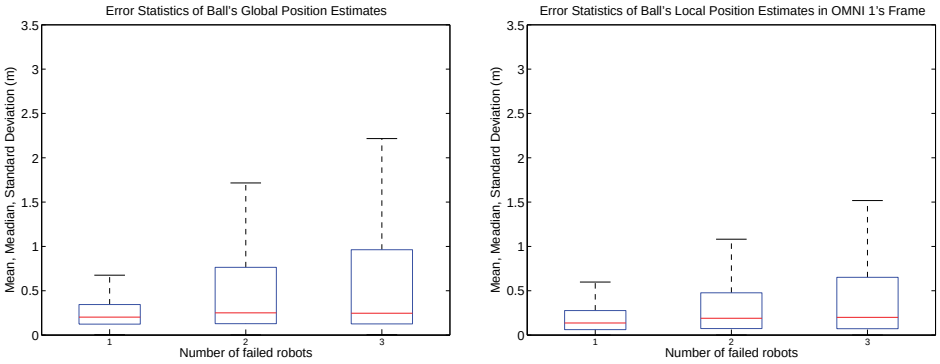
3 Failed Robots: OMNI2 loses camera vision from $\sim 80^{\text{th}}$s up till the end, OMNI3 loses camera vision from $\sim 85^{\text{th}}$s up till the end and OMNI4 loses camera vision from $\sim 92^{\text{nd}}$s up till the end.

Table 6.4 and the plots in Figures 6.5 and 6.6 present the results of the PF-UCLT algorithm and the O-MMLG method applied in this scenario.

The evolution of all the errors in this scenario's experiments are very similar to those in the case of permanent communication failure, except for the fact that in this case, the mean/median errors of the robots' positions are slightly higher. This is because, in the case of vision failure, the unified CP algorithms still try to localize the 'faulty' robot (i.e, the robot which loses camera vision) based only on its odometry measurements (which are often very poor, e.g, OMNI4's highly noisy odometry deteriorated the mean of its position error upto $\sim$ 3m in both PF-UCLT and O-MMLG methods). However, in the case of permanent communication failure, the 'disconnected' robots are simply not considered for localization after they lose communication.



PF-UCLT



O-MMLG

Figure 6.5: The box plots in this figure present the statistical estimates of the ball's global (and local in OMNI1's frame) position estimation errors for the PF-UCLT and the O-MMLG experiments performed in the scenario of permanent vision failure. The X-axis represents the three experimental situations, described in the sub-section 6.2.3, of this scenario.

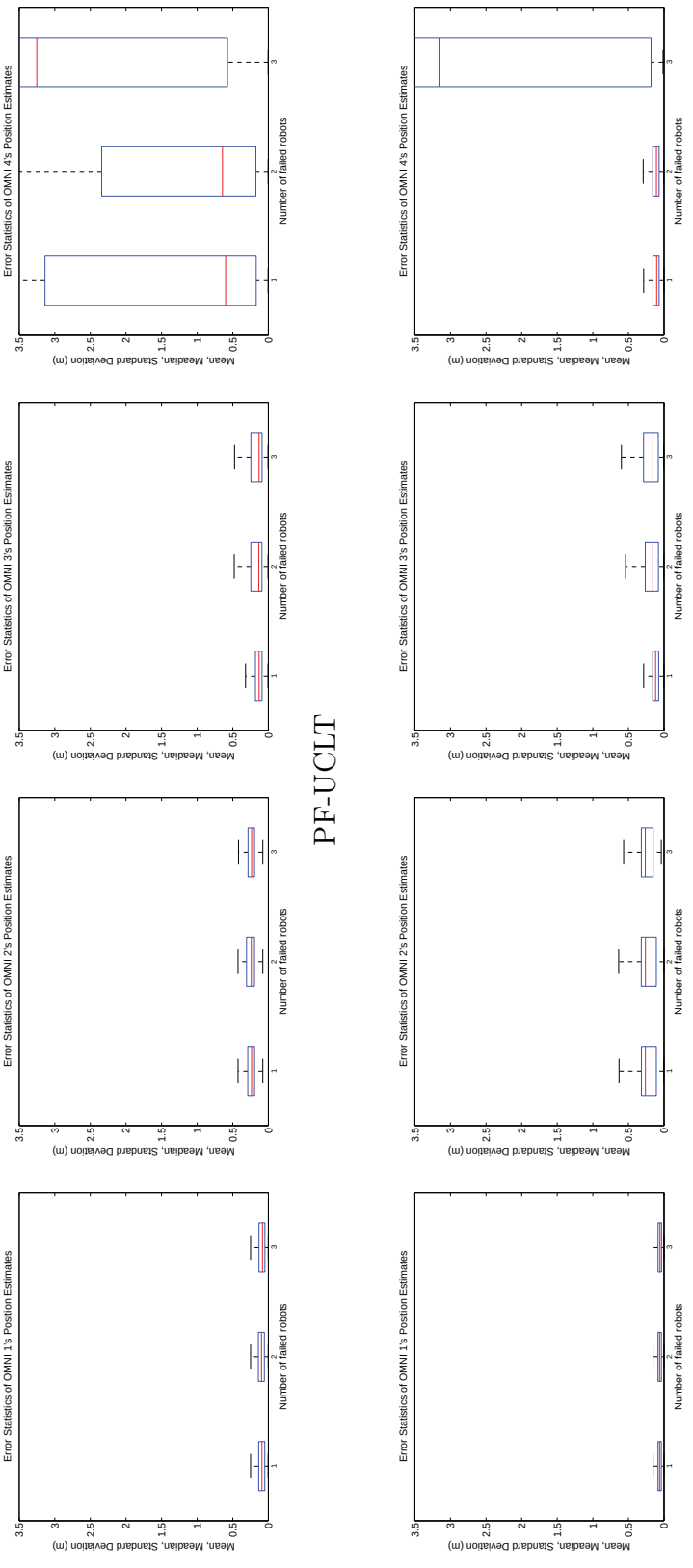


Figure 6.6: The box plots in this figure present the statistical estimates of all the robots' position estimation errors for the PF-UCLT and the O-MMLG experiments performed in the scenario of permanent vision failure. The X-axis represents the three experimental situations, described in the sub-section 6.2.3, of this scenario.

		PF-UCLT			O-MMLG		
Entity	Failed Robot(s)	Mean (m)	Median (m)	Std.dev. (m)	Mean (m)	Median (m)	Std.dev. (m)
OMNI1	OMNI2	0.096	0.090	0.056	0.064	0.062	0.035
	OMNI2,3	0.100	0.097	0.055	0.064	0.063	0.034
	OMNI2,3,4	0.093	0.083	0.054	0.062	0.057	0.035
OMNI2	OMNI2	0.248	0.236	0.090	0.298	0.261	0.210
	OMNI2,3	0.257	0.240	0.110	0.297	0.261	0.210
	OMNI2,3,4	0.243	0.236	0.075	0.271	0.263	0.151
OMNI3	OMNI2	0.152	0.130	0.102	0.123	0.119	0.068
	OMNI2,3	0.208	0.134	0.194	0.233	0.157	0.232
	OMNI2,3,4	0.191	0.131	0.167	0.245	0.156	0.242
OMNI4	OMNI2	1.547	0.601	1.616	0.201	0.104	0.434
	OMNI2,3	1.400	0.645	1.513	0.187	0.107	0.393
	OMNI2,3,4	3.052	3.253	2.000	3.344	3.160	2.787
Ball Global Position	OMNI2	0.657	0.365	0.697	0.334	0.202	0.413
	OMNI2,3	0.904	0.628	0.778	0.606	0.251	0.744
	OMNI2,3,4	0.959	0.597	0.870	0.661	0.247	0.788
Ball Local Position	OMNI2	0.429	0.226	0.531	0.244	0.137	0.344
	OMNI2,3	0.583	0.326	0.608	0.403	0.190	0.537
	OMNI2,3,4	0.598	0.328	0.647	0.464	0.200	0.590

Table 6.4: Error statistics in the situation of permanent vision failure.

6.2.4 Temporary Vision Failure

This sub-section describes the following 3 experimental situations (in increasing order of the number of failed robots) under the scenario of a temporary vision failure.

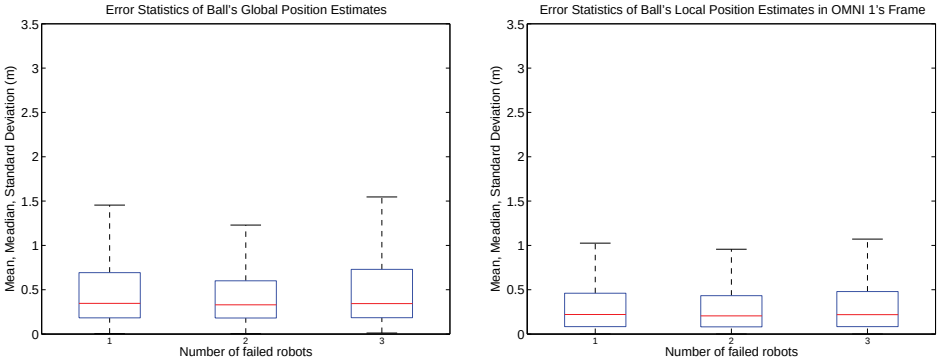
1 Failed Robot: OMNI2 loses camera vision from $\sim 80^{\text{th}}$s up to $\sim 120^{\text{th}}$s.

2 Failed Robots: OMNI2 loses camera vision from $\sim 80^{\text{th}}$s up to $\sim 112^{\text{nd}}$s and OMNI3 loses camera vision from $\sim 85^{\text{th}}$s up to $\sim 120^{\text{th}}$s.

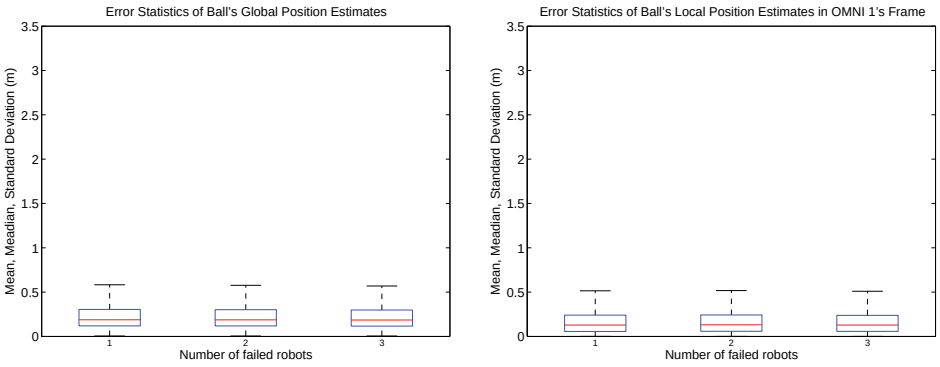
3 Failed Robots: OMNI2 loses camera vision from $\sim 80^{\text{th}}$s up to $\sim 112^{\text{nd}}$s, OMNI3 loses camera vision from $\sim 85^{\text{th}}$s up to $\sim 120^{\text{th}}$s and OMNI4 loses camera vision from $\sim 92^{\text{nd}}$s up to $\sim 128^{\text{th}}$s.

Table 6.5 and the plots in Figures 6.7 and 6.8 present the results of the PF-UCLT algorithm and the O-MMLG method applied in this scenario.

The mean/median errors of the experiments in this case reflect the property of the unified CP algorithms to robustly handle the temporary vision failures in the participating robots. All the errors are similar to those in the case of temporary communication failure and even lower for some entities, e.g, mean errors of OMNI2 (O-MMLG), OMNI3 (PF-UCLT) and the ball (both PF-UCLT and O-MMLG). This highlights the ability of our methods to recover 'faulty' robots' localization and object tracking estimates, once their vision system is restored.



PF-UCLT



O-MMLG

Figure 6.7: The box plots in this figure present the statistical estimates of the ball's global (and local in OMNI1's frame) position estimation errors for the PF-UCLT and the O-MMLG experiments performed in the scenario of temporary vision failure. The X-axis represents the three experimental situations, described in the sub-section 6.2.4, of this scenario.

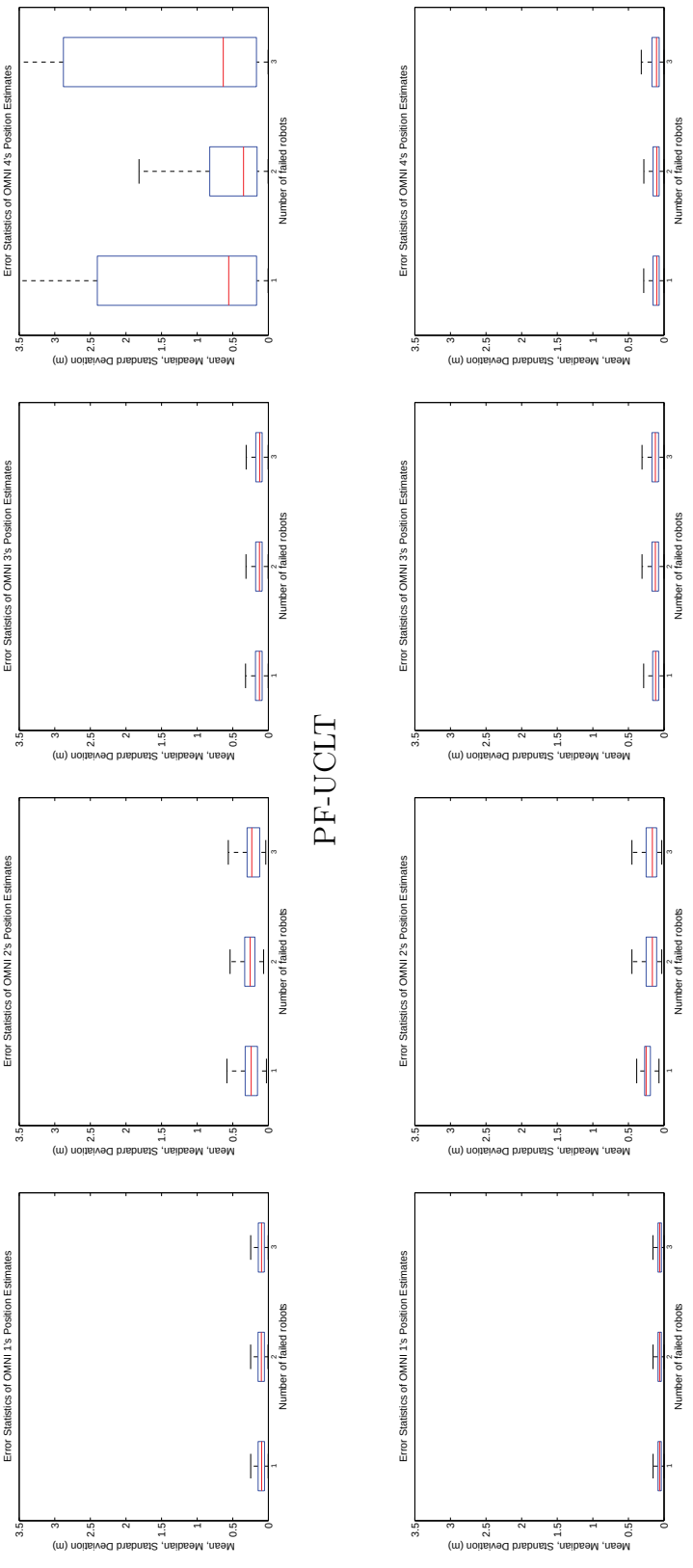


Figure 6.8: The box plots in this figure present the statistical estimates of all the robots' position estimation errors for the PF-UCLT and the O-MMLG experiments performed in the scenario of temporary vision failure. The X-axis represents the three experimental situations, described in the sub-section 6.2.4, of this scenario.

		PF-UCLT			O-MMLG		
Entity	Failed Robot(s)	Mean (m)	Median (m)	Std.dev. (m)	Mean (m)	Median (m)	Std.dev. (m)
OMNI1	OMNI2	0.100	0.094	0.055	0.064	0.061	0.035
	OMNI2,3	0.102	0.099	0.056	0.064	0.061	0.035
	OMNI2,3,4	0.100	0.096	0.055	0.063	0.061	0.035
OMNI2	OMNI2	0.259	0.241	0.148	0.224	0.252	0.077
	OMNI2,3	0.270	0.255	0.135	0.176	0.163	0.084
	OMNI2,3,4	0.235	0.232	0.133	0.176	0.163	0.084
OMNI3	OMNI2	0.148	0.125	0.106	0.123	0.118	0.068
	OMNI2,3	0.151	0.125	0.111	0.143	0.122	0.113
	OMNI2,3,4	0.150	0.121	0.112	0.143	0.122	0.113
OMNI4	OMNI2	1.307	0.557	1.400	0.184	0.102	0.376
	OMNI2,3	0.700	0.348	0.855	0.183	0.102	0.376
	OMNI2,3,4	1.519	0.633	1.580	0.188	0.105	0.375
Ball Global Position	OMNI2	0.582	0.347	0.626	0.282	0.188	0.333
	OMNI2,3	0.525	0.329	0.549	0.297	0.187	0.388
	OMNI2,3,4	0.607	0.343	0.653	0.286	0.185	0.362
Ball Local Position	OMNI2	0.383	0.221	0.468	0.202	0.129	0.272
	OMNI2,3	0.368	0.205	0.451	0.203	0.131	0.273
	OMNI2,3,4	0.388	0.219	0.471	0.197	0.128	0.261

Table 6.5: Error statistics in the situation of temporary vision failure.

6.3 Summary

In this chapter we presented a case study performing a feature comparison of all the CP algorithms developed in this thesis and a quantitative comparison of the two unified CP methods, O-MMLG and PF-UCLT, presented in Chapter 5. In the feature comparison, we outlined the specific requirements of the CP methods that enables a reader to chose a particular method for a certain application scenario. The quantitative analysis explores the behavior of the unified CP methods in communication and vision sensor failure situations. It highlights that the methods are robust to such failures by not allowing the faulty robots (robots that have lost communication or vision) affect the faultless robots' (robots that have communication and vision sensors intact) localization estimates and the tracked object's position estimates. If the object remains unobserved by the faultless robots, its tracked

position estimate's error grows, however, the tracked object's correct position estimate is recovered as soon as it becomes visible again to the faultless robots.

Chapter 7

Case Study 2: Cooperative Perception With Closed Loop Formation Control

7.1 Introduction

Most of the past and current work on motion coordination of multiple (possibly heterogeneous) vehicles focuses on controlling a vehicle formation with a given nominal geometry and a pre-determined trajectory or a static destination location, possibly compliant with the presence of obstacles on the formation trajectory. Such methods typically:

- assume full knowledge of the formation state, expressed as the relative distances and bearings among all the vehicles, and/or
- rely on local memory-less interactions, often jeopardizing global formation stability.

A vehicle formation is supposed to serve one or more mission objectives [62]. One such interesting case concerns localizing or tracking relevant objects, here and henceforth denominated as targets. Available formation control methods often give little relevance to the requirements imposed by target tracking to the formation geometry, so as to improve the target detection and the tracking quality (e.g., accuracy). Active cooperative perception methods in sensor and robot networks [2] concern precisely this problem: how to actively move mobile sensors so as to improve the accuracy of target detection by the network, as the result of (spatially and temporally) fusing the information from all the static and

mobile sensors which observe the target during a step sequence. In this case study we propose a solution for the "target localization and tracking by a vehicle formation" problem by integrating the following modules:

- the Algorithm 3.1 developed in Chapter 3 for cooperative target tracking based on a particle filter (PF) which estimates the target's position and velocity;

- a formation controller (FC)[1], with the control objective of minimizing the uncertainty of a cooperatively target while simultaneously achieving other criteria, e.g, keeping a pre-set distance to the target and/or avoid collisions between teammates in the formation while tracking the target. This was developed as the subject of an another PhD thesis within the frame of a joint national project[2] between ISR/IST and INESC-TEC .

Therefore the solution integrates two basic modules: i) a controller and ii) an estimator.

The control module consists of a nonlinear model predictive formation controller (NMPFC). Recent approaches for active cooperative target tracking by a robot team formation such as [2] rely on computationally heavy optimization process. By introducing the Gauss-Seidel relaxation in an iterative algorithm to detect the next best sensing location for the mobile sensors, the authors in [2] achieve a linearly growing computational complexity over methods like grid-based exhaustive search which have similar tracking accuracy but where the complexity grows exponentially with the number of sensors. The novelty in our approach of integrating the control and estimator modules to achieve a formation that minimizes the joint uncertainty covariance of the tracked target, lies in the fact that the control module of each robot performs an optimization over an already fused target's posterior which makes the computational complexity of the optimization process constant with respect to the number of mobile sensors in the team. Furthermore the decoupling of the optimization problem from the cooperative target estimation makes the approach more reliable in case of individual sensor or inter robot communication failures.

[1]The controller was developed and implemented by Tiago Nascimento et al. in [63] as one of the 'Task Modules' of the project PCMMC: Perception Driven Coordinated Multi-Robot Motion Control (Reference No: FCT PTDC/EEA-CRO/100692/2008). For a detailed description of the controller, please see [64], PhD thesis of Tiago Nascimento. In this case study chapter, excerpts from [64] are used to briefly describe the formation controller.

[2]PCMMC project homepage:
http://welcome.isr.ist.utl.pt/project/index.asp?accao=showproject&id_project=157

On each robot, the estimator module acts as a feedback module providing the position and velocity estimates of the tracked target to the control module which in turn uses these estimates as well as the teammates' positions to generate velocity set points for that robot.

The rest of the chapter is organized as follows. After a brief overview of both the control and the estimator module in Section 7.2, we describe their integration in Section 7.3. This is followed by the implementation details of this approach on our testbed and the experimental results in Section 7.4. A summary of this chapter is presented in Section 7.5.

7.2 The Control and Estimator Modules

7.2.1 Control Module

The NMPFC, acting as the control module in this work, has its roots in the non-linear model predictive controller (NMPC) developed and implemented in [63]. NMPC has a partially distributed architecture where each robot calculates its own control inputs by solving its own optimization problem having a central unit only as a communication bridge. In the fully distributed architecture of the NMPFC, the communication is performed by a real-time data base (RTDB) system [65]. This enables the robots to be communication failure tolerant ensuring the formation's stability. Furthermore, even in the rare case of a communication failure, the robots use their predictive open-loop strategy to determine their teammates' states making the NMPFC even more robust.

The NMPFC's ability to create and maintain a formation is due to the fact that the cost functions used by the controllers of each robot in the team are coupled. This coupling occurs when the teammates' states (position and velocity) are used in the cost function of each robot's controller to penalize the formation geometry or the deviation from the desired objective thus the actions of each robot affect every other teammate. The NMPFC iterates the following two components:

- **Optimizer** - Uses an online numeric minimization method to optimize the cost function and generate the control signals. The resilient propagation (RPROP) method, that is used here, guaranties quick convergence.

- **Predictor** - Performs the state evolution of a particular entity based on its predefined model. The entity could be the robot itself, its teammates in the formation or another object in the environment essential to the formation's objective such as a static obstacle or a moving target.

The optimizer, in each iteration of the controller, after receiving the predicted states of the entities in the formation from the predictor, provides the control signal back to the predictor to perform the formation state evolution for a fixed number of prediction horizons and generate a cost value with respect to that control signal. These iterations continue until the minimum of the cost function is reached.

7.2.2 Estimator module

The estimator module is a cooperative target estimator (CTE) which consists of the PF-based cooperative tracker algorithm (Algorithm 3.1) developed and presented in Chapter 3. Instead of the traditional update step in the CTE's PF [43], a fusion step is performed. The fusion involves the communication of the target observation measurements in the global frame, observation measurement confidences and the self-localization confidences among all the robots in the team. Through a process of confidence-based selection, each robot performs an update over its particle set in a way such that the information from all its teammates is accounted for without getting corrupted by their poor self-localization or target observation. The CTE functions in a decentralized manner enabling each robot in the team to run its own instance of the CTE and therefore suitable for integration with the NMPFC.

7.3 Module Integration

The central objective of the formation control in this work is to minimize the total uncertainty of the target's cooperative estimate as perceived by the formation which is achieved by integrating the control and the estimator modules. It is done by formulating the NMPFC's cost function incorporating the fused target estimate obtained from the CTE along with the other formation criteria parameters such as inter-robot distances, distance to the target, etc.

A flow diagram describing the integration of the control and the estimator modules is presented in Figure 7.1. Each robot runs an instance of the NMPFC and the CTE. The CTE, represented as a single block in Figure 7.1, communicates to the NMPFC the cooperatively estimated target's position, covariance matrix and the target's velocity. The

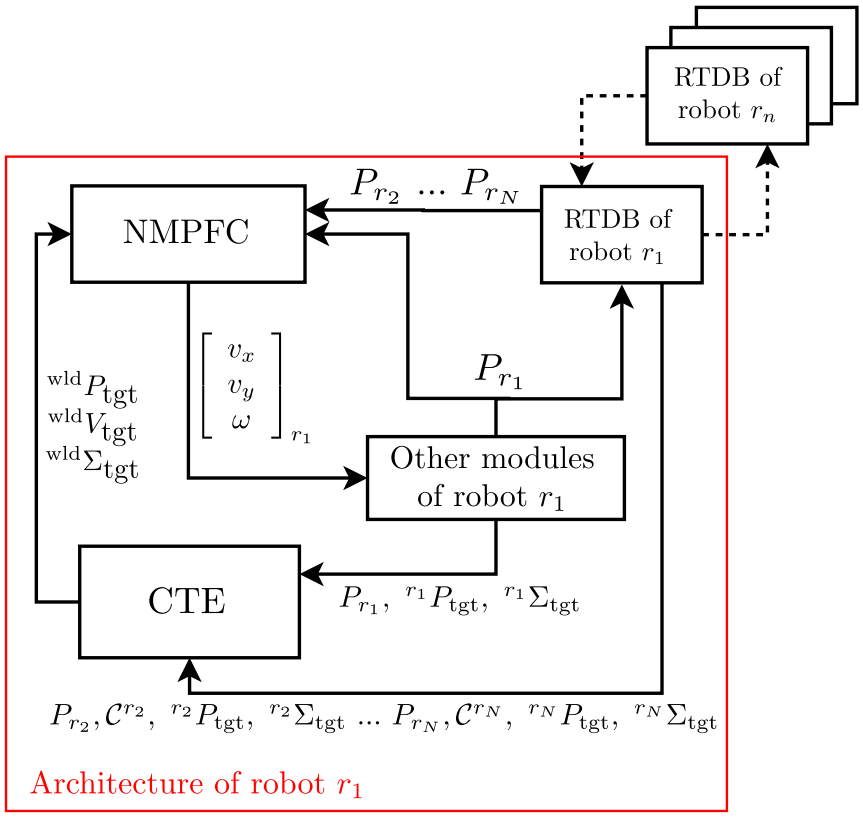


Figure 7.1: The formation control loop describing the control-estimator module integration. The above flow diagram represents the architecture of robot r_1 in a team of N robots where the n^{th} robot is denoted by r_n. P_{r_n} denotes the robot r_n's world frame pose (position + orientation) as obtained by its self-localization mechanism (implemented separately from the formation control loop) and $\mathcal{C}^{r_n}$ denotes its localization confidence. ${}^{r_n}P_{\text{tgt}}$ denotes the target observation measurement made by the robot r_n in the world frame and ${}^{r_n}\Sigma_{\text{tgt}}$ denotes the covariance matrix of zero-mean Gaussian noise associated with this measurement. ${}^{\text{wld}}P_{\text{tgt}}$ and ${}^{\text{wld}}V_{\text{tgt}}$ denote the target's cooperatively estimated world frame position and velocity, respectively. ${}^{\text{wld}}\Sigma_{\text{tgt}}$ denotes the target's cooperatively estimated position covariance matrix. The vector $\begin{bmatrix} v_x & v_y & \omega \end{bmatrix}^{\top}_{r_1}$ denotes the velocity set points for the robot r_1, which is the output of the NMPFC at that robot. The block named 'other modules of robot r_1' denotes that robot's low level control and sensor units, e.g, robot wheel controller and target detector (using camera images). This flow diagram is reprinted from page 45 of [64] with modifications in the variable nomenclature.

NMPFC's cost function[3] (for the details of the cost function please refer to [64]) at each robot also requires the teammates' positions. This is achieved through an additional layer of the RTDB's shared memory where all the robots write their own pose estimates (positions and orientation) and read their teammates' poses.

The FC's cost function involves seven terms (each term has an associated penalization weight) which are described as follows.

- The first term concerns the target's fused uncertainty matrix determinant penalization which brings the robots in a geometric configuration w.r.t. the target more adequate to reduce the uncertainty of the target observation.
- The second term prevents the robots from colliding with the target.
- The third term keeps the robots' poses oriented towards the target position.
- The fourth term orients the robots' velocity direction w.r.t. that of the target according to a predefined orientation.
- The fifth term prevents any inter-robot collisions.
- The sixth prevents the robots from colliding with the static obstacles in the environment.
- The seventh and last term is the control effort penalization which restricts significant changes and oscillations in the control signal.

The first term, involving the target's fused uncertainty matrix determinant penalization, is initialized with the fused target position covariance matrix received from the CTE. Subsequently, over the rest of the prediction horizons, NMPFC's predictor uses a predefined covariance evolution model[3], to predict the target position's covariance matrix evolution for the iterative minimization of the full cost function. Once the minimum is reached, the NMPFC sends the velocity signals to the robot's wheel controllers so that the robot reaches the next best location maximizing the cooperative perception of the target while maintaining a preset threshold distance between teammates, between itself and the target as well as between itself and the obstacles in the environment.

[3]For the sake of brevity and to avoid notational confusion in this thesis, the controller's cost function and the target's position covariance evolution model is not presented in this chapter. Please refer to pages 43–56 of Tiago Nascimento's PhD thesis [64] for a complete description of these.

7.4 Implementation, Experiments and Results

The formation control's implementation was performed on two separate RoboCup soccer middle size league (MSL) teams: i) 5dpo and ii) SocRob, both of which are described in Appendix A. The tracked target was a standard FIFA size 5 soccer ball for both teams. Implementation details, simulations and real robot results with an analysis are presented further.

7.4.1 Implementation

The FC's cost function, as mentioned in the previous section, has several terms with changeable penalization weights. In order to understand the behavior and influence of the covariance term (and therefore of the Algorithm 3.1) that minimizes the determinant of the target's fused uncertainty covariance matrix, the following three situations were tested both in simulations and in real robots:

1. Penalizing the determinant of the target position's uncertainty covariance matrix (first term of the FC's cost function) and the control effort (seventh term of the FC's cost function). This situation is further referred to as '*only target covariance*' situation.

2. Penalizing the distances between the robots and the target (second term of the FC's cost function), the collision between the teammates (fifth term of the FC's cost function) and the control effort (seventh term of the FC's cost function). This situation is further referred to as '*only mates*' situation.

3. Penalizing every term in the FC's cost function by giving a positive penalization weight to all the terms. This situation is further referred to as '*all terms*' situation.

Simulation experiments were conducted with both the 5dpo and the SocRob robot soccer teams for all the three situations described above. For each team, we conducted two separate sets of experiments. The first set involved 2 robots while the second involved 3 robots. Results of all these simulations are presented. The maximum allowed velocity in the 5dpo simulations was 1.4m/s while in the SocRob simulations was 0.5m/s due to the differences in the dynamics of the robots. Each team uses its own target position covariance evolution model[3] [64]. The 5dpo robots' simulations were made using the software SimTwo[4]

[4]SimTwo: http://paginas.fe.up.pt/ paco/wiki/index.php?n=Main.SimTwo

and that of the SocRob robots' were made using the software Webots simulator[5]. Robots' trajectory plots and target's fused position covariance matrix determinant plots of selected simulation experiments are presented in Figure 7.2. Table 7.1 presents the results for both the team's first set of the experiments involving 2 robots and Table 7.2 presents the results for both the team's second set of experiments involving 3 robots.

Real robot experiments were performed using a team of 2 SocRob robots for all the three situations mentioned previously. The video[6] accompanying this chapter shows the simulation and real robot experiments' footage where the trajectory for each robot and the formation can be visualized. Results of the real robot experiments are presented in Table 7.4.

7.4.2 Simulation Results

5dpo - 2 robots case	
Situation	$\lvert {}^{\text{wld}}\Sigma_{\text{tgt}} \rvert$ (m^2)
Only target covariance	0.2252
Only mates	0.3145
All terms	0.2201
SocRob - 2 robots case	
Situation	$\lvert {}^{\text{wld}}\Sigma_{\text{tgt}} \rvert$ (m^2)
Only target covariance	0.3356
Only mates	0.4205
All terms	0.3415

Table 7.1: Results of all three situations in the simulation experiments with 2 5dpo robots and 2 SocRob robots. $\lvert {}^{\text{wld}}\Sigma_{\text{tgt}} \rvert$ denotes the target's cooperatively estimated position covariance matrix determinant's final value.

Tables 7.1 and 7.2 show, for all the three experimental situations described previously, the final values of the target's cooperatively estimated position covariance matrix determi-

[5]Webots: http://www.cyberbotics.com/overview

[6]Video link of Case Study 2's experiments.
http://users.isr.ist.utl.pt/∼aahmad/PhDThesisVideos/
Chap_7_Perception_driven_formation/Perception_Formation.mpg

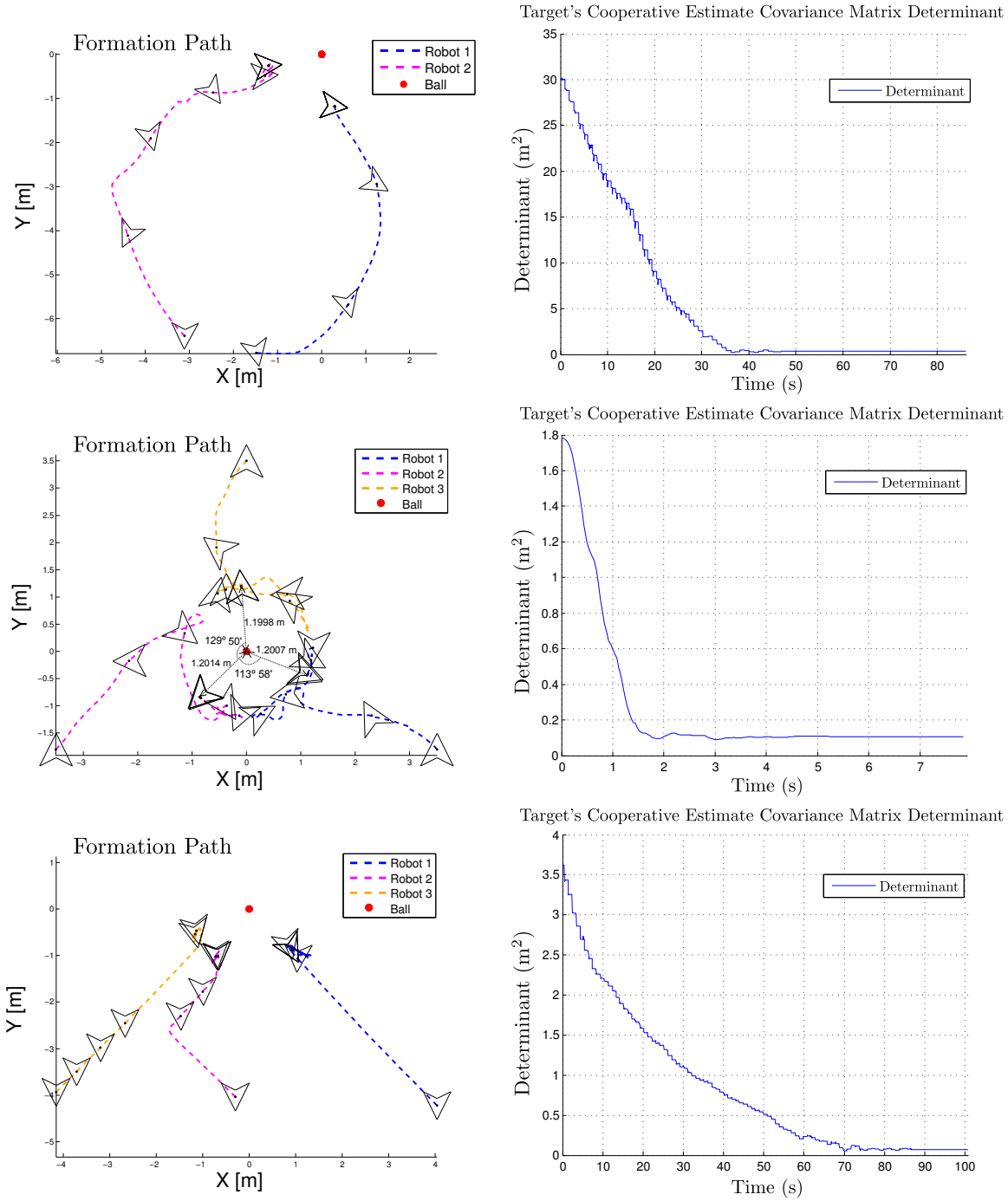


Figure 7.2: The plots in the left column show the robots' trajectories in the formation and their final poses during the simulation experiments, where as the plots in the right column, next to the trajectories, show the corresponding evolution of the cooperatively estimated target position's covariance matrix determinant. The plots in the first row are for SocRob's 2 robot case for the '*only target covariance*' situation. The second row plots are for 5DPO's 3 robots case for the '*all terms*' situation. The third row plots are for SocRob's 3 robot case for the '*only target covariance*' situation. These plots are reprinted from [64].

5dpo - 3 robots case	
Situation	$\lvert{}^{\text{wld}}\Sigma_{\text{tgt}}\rvert$ (m^2)
Only target covariance	0.1017
Only mates	0.1095
All terms	0.1074
SocRob - 3 robots case	
Situation	$\lvert{}^{\text{wld}}\Sigma_{\text{tgt}}\rvert$ (m^2)
Only target covariance	0.0742
Only mates	0.0646
All terms	0.0924

Table 7.2: Results of all three situations in the simulation experiments with 3 5dpo robots and 3 SocRob robots. $\lvert{}^{\text{wld}}\Sigma_{\text{tgt}}\rvert$ denotes the target's cooperatively estimated position covariance matrix determinant's final value.

nant (denoted by $\lvert{}^{\text{wld}}\Sigma_{\text{tgt}}\rvert$) after the formation achieves convergence. Table 7.3 summarizes the penalization weights used for each term of the FC's cost function during the experiments. It should be noted that the SocRob's target position covariance model is derived from its observation model (2.1) and hence it naturally finds the best position w.r.t. the target for minimizing $\lvert{}^{\text{wld}}\Sigma_{\text{tgt}}\rvert$, whereas the 5dpo's model is built empirically [64] for which the minimization would theoretically occur when the robots' and target's positions coincide. Therefore in the FC's cost function, the penalization weight for the second term, which prevents the robot-target collision, is always a positive value in the case of 5dpo robots but not for the SocRob robots.

From the results of all the three experimental situations in the simulations for both the 5dpo and the SocRob robots, we infer that the final minimized value of $\lvert{}^{\text{wld}}\Sigma_{\text{tgt}}\rvert$ achieved in the 3 robots case was lower than in the 2 robots case. This highlights that the PF-based cooperative tracker (Algorithm 3.1) achieves better accuracy of the target perception with the increase in the number of robots.

In most cases (2 robot's case for both the 5dpo and the SocRob robots and the 3 robots case for the 5dpo robots), we observe that the '*only mates*' situation results in a higher value of $\lvert{}^{\text{wld}}\Sigma_{\text{tgt}}\rvert$ as compared to the other two situations '*only target covariance*' and '*all terms*'. This signifies that the FC's objective to minimize the target's cooperative

	Only target covariance		***Only mates***		***All terms***	
FC's Cost function term number	5dpo	SocRob	5dpo	SocRob	5dpo	SocRob
1	3000	300	0	0	3000	300
2	1500	0	1500	1500	1500	1500
3	0	0	0	0	300	200
4	0	0	0	0	100	100
5	0	0	500	600	500	600
6	0	0	0	0	600	600
7	5	100	5	100	5	100

Table 7.3: The table presents the penalization weight values for each term of the FC's cost function. These penalization weights were used for both the 2 and 3-robots cases and for the simulation and real robots experiments. These values were automatically found using a gradient search method. It finds the set of penalization weights in a given experimental situation which minimizes an objective function. The objective function's arguments are the set of penalization weights, the convergence time taken by the robots and the minimized value of the FC's cost function. This was developed by Tiago Nascimento et al. and is described in the pages 61–63 of [64].

perception is better achieved only when the FC's cost function's target position covariance term is enabled (has a positive penalization value). It is also notable that the value of $|^{\text{wld}}\Sigma_{\text{tgt}}|$ is lower in the '*only target covariance*' situation as compared to the '*all terms*' situation. This is because when all the terms in the FC's cost function are penalized, the robots tradeoff the better cooperative perception of the target in order to improve on other formation criteria, e.g, maintaining a predefined inter-robot distance and predefined threshold distance to the target.

An exception is identified in the 3 robots case for the SocRob robots. Here the value of $|^{\text{wld}}\Sigma_{\text{tgt}}|$ is higher in the '*only target covariance*' situation as compared to the '*only mates*' situation. The reason for it is as follows. Due to the covariance merging method using Smith and Cheeseman's formulation [66] (pages 50-51 and Appendix A of [64]) in these experiments, theoretically two global minima would exist for a formation of 3 robots in

order to achieve convergence in the '*only target covariance*' situation. In one of these global minima, 2 of the 3 robots' converged positions will coincide with each other whereas in the second global minima, these positions will be on the diametrically opposite sides of the target. Since, in the '*only target covariance*' situation, the term concerning the inter-robot collisions is not penalized, the robots tried to achieve convergence while colliding with each other and thus never actually attained the theoretically correct convergence position. This led to a higher value of $|^{\text{wld}}\Sigma_{\text{tgt}}|$. This can also be graphically visualized in the left column third row trajectory plot of Figure 7.2 where the situation of '*only target covariance*' for the 3 robots in the SocRob's case is presented. However, in the 3 robots case for the 5dpo robots, a lower value of $|^{\text{wld}}\Sigma_{\text{tgt}}|$ was achieved in the '*only target covariance*' situation as compared to the '*only mates*' situation because the robots converged in the diametrically opposite sides of the target.

7.4.3 Real Robot Results

Situation	$\lvert^{\text{wld}}\Sigma_{\text{tgt}}\rvert$ (m^2)
Only Target Covariance	0.687
Only Mates	0.703
All terms	0.233

Table 7.4: Results of all the three situations for SocRob's 2 real robots case. $|^{\text{wld}}\Sigma_{\text{tgt}}|$ denotes the target's cooperatively estimated position covariance matrix determinant's final value.

The video accompanying this chapter also presents the footage of the 2 SocRob real robots' experiments. The penalization weights for each term in the FC's cost function were the same as in the simulations. Table 7.4 shows the value of $|^{\text{wld}}\Sigma_{\text{tgt}}|$ after the formation's convergence. It should be noted that in the experiment situation '*all terms*' for the real robot's case, a lower value of $|^{\text{wld}}\Sigma_{\text{tgt}}|$ is obtained as compared to the other two situations. This is because in the other two situations, the chances for the formation to get stuck in a local minimum are higher, hence introducing the other terms in the cost function such as teammate avoidance and orientation towards the target helps converging the formation to a global minima.

7.5 Summary

A method for perception-driven multi-robot formation control was proposed and implemented in this case study. Particle filter-based cooperative object tracking (Algorithm 3.1), developed in the Chapter 3, was applied as a feedback module in a vehicle formation control loop. The simulation and real robot results demonstrate the success of its implementation on two different kinds of robot teams. The penalization weight-based minimization of the formation controller's cost function lets us control different objectives like better target perception, inter-robot/target-robot collision avoidance, etc., while creating and maintaining the robot-team formation. We demonstrated the applicability of Algorithm 3.1 as well as the formation control loop by achieving formation convergence in robots with different dynamics and target observation models.

7.6 Related Publications

The work presented in this chapter was accepted as a full length-article [11] to be published in the proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013), Karlsruhe, Germany (submitted: September 2012, accepted: January, 2013).

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we presented novel techniques for cooperative perception methods to address the issues, such as, sensor fusion, disagreement between sensors, localization errors of the mobile platforms on which the sensors are mounted, and occlusions, within an integrated Bayesian framework. Based on the desired objective and the given scenario, in the context of cooperative perception, we proposed the following methods:

- For a scenario where the aim is to cooperatively track a moving object through a team of mobile robots, equipped with sensors, and the pose of the robots is given through a self-localization system running on them, we presented a particle filter-based multi-robot cooperative object tracking algorithm (PF-MCOT). Here, teammates share object observation measurements in the global frame, their measurement confidence and their self-localization confidence. By incorporating these confidences while fusing the teammates' measurements, PF-MCOT handles sensor disagreement, occurring due to poor localization of the robots or poor observations made by them, within an integrated framework.

- For a scenario where the aim is to cooperatively localize a team of robots, given that there is a visually shared object being tracked by the robots, we presented a PF-based multi-robot cooperative robot localization algorithm (PF-MCRL). Here, the robots communicate to their teammates the global frame tracked position of the visually shared object, a confidence on that position and their self-localization confidence. If a team robot detects that its self-localization confidence is below a certain threshold (i.e, it is lost), its PF-MCRL particles are reset according to the visually shared

object's global position received from its teammates as well as its own local frame information regarding that object. This enables the lost robots to regain their correct localization.

- For a scenario where the aim is to cooperatively localize a team of robots as well as cooperatively track a moving object through those robots, given that the environment consists of fixed and known static landmarks and the robots do not measure inter-robot distances, we presented the following unified approaches.

  - Multi-robot moving landmark graph optimization (O-MMLG) method. Here, we represented the above-mentioned problem through a pose graph and considered the tracked object as a moving landmark, in addition to the known and static landmarks. The nodes of this graph represent the states (poses of all the robots and the position of the object) to be estimated whereas the edges represent the measurements made by the robots. A least squares error function is obtained from the pose graph and solved using a state-of-the-art non-linear least squares solver [33]. The solution is the configuration of the nodes (and therefore all the the states) that best describe all the measurements. This is an offline method where all the measurement data is processed in a batch.
  - PF-based unified cooperative robot localization and object tracking (PF-UCLT) algorithm. Here, at every timestep, each robot shares its odometry and observation measurements with its teammates. It is an online method where the filter, at each robot, estimates its own pose, the pose of all its teammates and the position of the tracked object. We showed that our PF-UCLT algorithm requires a number of particles that grows linearly with the number of robots, instead of exponentially, as would otherwise be required [56] by a standard PF solution to this problem. This was achieved by utilizing the properties of conditional/mutual independence of some of the involved variables.

PF-MCOT and PF-MCRL are designed to operate in an online manner for realtime applications. We experimentally validated PF-MCOT in scenarios where a robot loses the vision of the tracked object, observes it poorly or is not well localized while observing the object. In all such situations the robot was successfully able to keep tracking the object due to the fusion of its measurements with that of its teammates. PF-MCRL was tested in situations where a robot, running this algorithm, was intentionally kidnapped. It was successfully able to re-localize itself if that robot and at least one of its teammates were simultaneously tracking a visually shared object.

Formal descriptions of all of the above-mentioned cooperative perception methods were presented. Experimental results for each method demonstrated its proof of concept, accuracy as well as scalability to higher number of robots. For a fair comparison of the unified methods O-MMLG and PF-UCLT, we used the exact same dataset for their experimental evaluation. Results show that PF-UCLT is slightly outperformed by O-MMLG due to the latter's iterative re-linearization and batch processing mode. However, the computational speed of the PF-UCLT method, in addition to its high accuracy, makes it suitable for realtime applications.

In the first case study, we presented a one-to-one comparison of the O-MMLG and PF-UCLT methods through a series of experiments conducted in various failure scenarios, such as, inter-robot communication failure and camera vision failure. Results demonstrated the methods' robustness to such failures. While the failed robots do not contribute much to the cooperative estimates of the other robots, they also did not degrade the functional robots' (the robots that did not fail during the experiment) estimates. In the second case study, we demonstrated the ability of PF-MCOT algorithm to integrate, as a feedback module, in a multi-robot formation control system that minimizes the uncerainty of the cooperatively tracked target.

8.2 Future Work

Despite the effectiveness of the proposed cooperative perception approaches in this thesis, relaxation of many subtle assumptions will extend the range and realism of application scenarios. This forms the central theme of our ongoing and future work. Some of those are enumerated as follows.

- Throughout this thesis, the tracked object was considered with known data association. This assumption is often void in many real scenarios. Furthermore there could be more than one object in the environment for a multirobot team to track, all with unknown data association, e.g, a team of robots interacting with multiple unknown people in a public place. Extending our cooperative perception approaches to include multiple tracked objects with unknown data association is identified as a major offshoot of this thesis.

- The traced object's motion model is one of the key factors in determining the accuracy and reliability of the tracker. In our work, we considered a constant velocity model with added zero mean Gaussian acceleration noise. Such a model is effective in many

situations. However, in order to model the random changes in the positions of the tracked object, a more complex and self-adapting model is required. This could even include predicting the motion model of the object itself, in addition to using that motion model to predict the next position of the object.

- The static and known landmark assumption is also not valid for many real scenarios. This implies extending our cooperative perception approaches in the direction of simultaneous localization and mapping (SLAM). SLAM is an extensively researched and well established topic. However, an approach which includes cooperative SLAM through a team of mobile robots while at the same time tracking moving objects in an unknown environment is an extension of the work accomplished in this thesis.

- Ongoing work includes extending our approaches to a heterogeneous team of robots. This requires incorporating different sensor and motion models into our integrated framework. This will further experimentally validate the effectiveness of the proposed cooperative perception approaches.

- Another significant offshoot of this thesis would be to extend the cooperative perception methods to multi-sensor human activity recognition. Consider a scenario where multiple robots cooperate to achieve cognitive tasks in cooperation with humans in their surrounding with on-board sensors on the robots in addition to a network of static sensors. This could involve multiple issues, e.g, continuously tracking persons and their activities without losing temporal information regarding that, ability to uniquely identify and track a person all the time. Cooperation among robots and network of static sensors to perform such tasks would significantly enhance the ability and effectiveness of each individual robot.

Appendix A

Soccer Robots Testbed and Datasets

The main testbed for the implementation of all the cooperative perception (CP) algorithms developed in this thesis is the RoboCup Middle Sized League (MSL)[1]. A team of robots, specifically designed to play robotic soccer and which participates regularly in the annual RoboCup world championship, was used. The ongoing 'SocRob[2]' project in the IRSLab of ISR, Lisbon holds the origin of these robots. Figure A.1 shows the team of robots placed on the robot soccer field of IRSLab.

Subsequent sections describe the robot's physical details, the vision system used on these robots and the dataset collected using them.

A.1 Soccer Robot Description

The robots used here are omnidirectional robotic soccer platforms as shown in Fig. A.2. It was developed by ISR/IST, in a joint venture with IdMind and ServiLog, two Portuguese SMEs. The most relevant details regarding the capabilities of its sensors and actuators are as follows:

[1]RoboCup MSL: http://wiki.robocup.org/wiki/Middle_Size_League
[2]SocRob Project: http://socrob.isr.ist.utl.pt/

Figure A.1: OMNI robots at the robot soccer field of IRSLab, ISR, Lisbon.

- Sensors:
  - A dioptric vision system consisting of a camera providing omnidirectional vision trough a fish-eye lens
  - Each of the robot's motors is coupled to a 500 CPR encoder for motor control and odometry.
  - A rate-gyroscope (XRS300EB) to improve self-localization.
  - Temperature sensors for detecting overheated motors/batteries.
  - A digital compass
- Actuators:
  - Each robot consists of three omni wheels, actuated by a MAXON DC motor (model RE35/118776), through a MAXON gear box (model 203118) with a

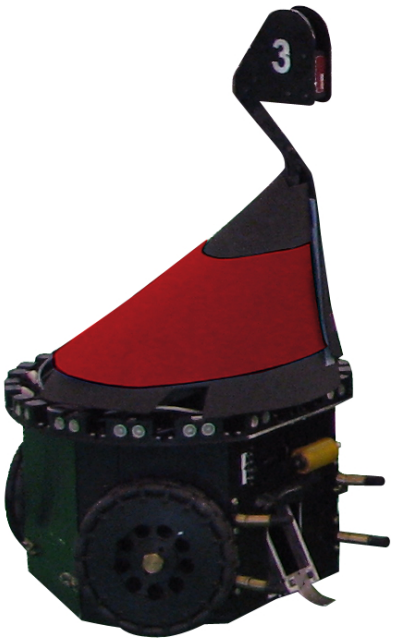


Figure A.2: OMNI (soccer robot) at ISR, Lisbon.

reduction of 21:1, providing the robotic platform with a maximum translational speed of approximately 3.5 m/s and maximum rotational speed of 20rad/s;

- In order to kick the soccer ball, an electromagnetic strength controlled kicker is installed;
- To aid in ball dribbling, a rolling drum is present near the kicker, with controllable rolling speed and elevation.

All of the above components are powered by two packs of 12V, 9Ah NiMH batteries per robot. In this robotic platform, the software architecture (which accounts for most of the required computational power) runs on a Sony Vaio laptop, equipped with an Intel Core i3 2.2GHz (quad core) CPU and 4GB of RAM, which is connected to the robot's sensors and actuators through plug-and-play connections (USB and Firewire).

A.2 Dioptric Vision System

The robot's vision system is based on an AVT Marlin F-033C firewire camera, which is equipped with a fish-eye lens providing a field-of-view of 185 °, facing downwards. This dioptric system endows the robot with omnidirectional vision, capable of detecting relevant

objects (such as the ball or obstacles) at a distance of up to 3.5 meters. This particular setup is also less sensitive to vibrations caused by the robot's motion than the previously used catadioptric system. An image acquired from this vision system is presented in Fig. A.3. The mathematical details for the projection model of this camera is presented in Chapter 2.

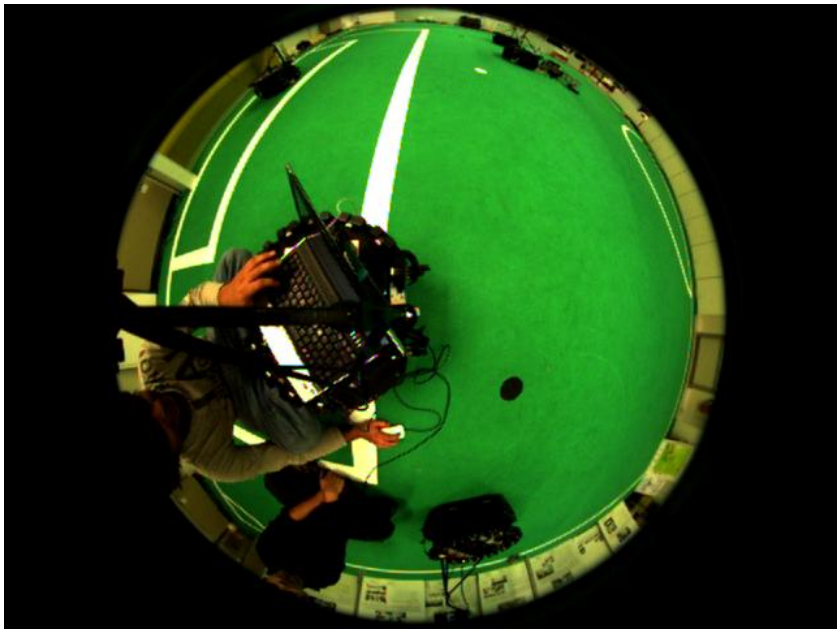


Figure A.3: Example image from the OMNI robot's on-board diotric vision system.

A.3 Datasets of Soccer Robots

In order to have a fair comparison of all the CP algorithms, datasets were collected using the 'SocRob' testbed described above. 4 OMNI robots, with colored markers placed on top of each of them, were used for the raw data collection. Two soccer balls (of different colors and FIFA standard size-5), connected to a thin string, were moved around in the IRSLab soccer field in a way such that their positions over time vary in 3D space. 10 distinct colored landmarks were placed at known locations on the field. The robots were manually manoeuvred in such a way that they do not collide with each other as well as come far and close to the ball positions at different times. A 'ZigBee'-based wireless localization module was placed on every robot to measure the inter-robot distances. A ground truth system (GTS), explained in detail in Appendix B, was used to save stereo image pairs while the raw data collection was done. Each robot's laptop, which was placed inside the

robot during the raw data collection, was used to log the following raw data with along with their associated timestamps (in microseconds):

1. Camera images from the dioptric vision system of the robots at $\sim$ 25 frames per second (FPS).
2. Odometry readings from the motors at $\sim$ 40 Hz.
3. Inter-robot distance measurements using 'ZigBee' modules at $\sim$ 4 Hz.

The GTS was connected to an external desktop computer which logged stereo image pairs at $\sim$ 20 FPS. All the robots' laptops and the GTS were synchronized using a time-synchronization software called 'chrony'. The raw data collection was performed 4 times, starting from 1 robot only and subsequently increasing the number of robots in the team each time. Hence 4 separate raw datsets were made and named accordingly. These are as follows.

- 1_robot_dataset
- 2_robots_dataset
- 3_robots_dataset
- 4_robots_dataset

Once the raw data was collected, 3D detection algorithm (as described in Chapter 2) for one of the balls (orange colored) and color based static landmark detection was performed on the images saved by each robot's vision system. Orange Ball's 3D position detection using its color and all the robot's 2D position (not the pose) detection using the colored marker placed on top of them were performed on the GTS' stereo pair images. Subsequently, the logs containing the following items were generated and used by all the CP algorithms:

1. The orange ball's 3D positions in each robot's local frame with timestamps.
2. All landmark's 2D positions (landmarks are fixed on the ground plane) in each robot's local frame with timestamps.
3. Each robot's odometry readings with timestamps.
4. The ground truth (GT) 3D positions of the orange ball and the 2D positions of all the robots with timestamps.

Note that the items enumerated above are less than all the measurement data that could be extracted from the raw data in the datasets made during the data collection. This is because the implementation of CP algorithms presented in this thesis made use of only the enumerated data herein.

The datasets described above, along with the measurement logs obtained from the raw data, are made publicly accessible[3]. It can be used by researchers in related areas to implement, test and verify various algorithms ranging from object detection, robot localization, cooperative methods for them, simultaneous localization and mapping, cooperative mapping and many more. This is identified as an additional contribution of this thesis.

A.4 5DPO Robot Soccer Platform

In case study 2, presented in Chapter 7, two separate soccer robot teams were used. The first being the 'SocRob' team as described previously here while the second is the robot soccer team: 5DPO[4] from the robotics laboratory of FEUP, Porto[5]

Figure A.4: 5DPO soccer robot at FEUP, Porto.

Figure A.4 shows one of the 5DPO robots. While its locomotion system is similar to the OMNI robots, i.e., consists of 3 omni wheels, there is a significant difference in its vision system. 5DPO has a cata-dioptric vision system which consists of a perspective lens-based camera facing upwards and pointing to the center of a parabolic mirror. The mirror faces downwards (towards the field) and is held in a fixed position w.r.t. the camera and the robot by supports connecting it to the robot's chassis.

[3]LRM Dataset download link: http://datasets.isr.ist.utl.pt/lrmdataset/4_Robots_DataSet/
[4]5DPO soccer team: http://paginas.fe.up.pt/∼robosoc/en/doku.php
[5]FEUP: http://sigarra.up.pt/feup/pt/web_page.inicial

Appendix B

Ground Truth System Details

In order to evaluate the results of all the cooperative perception (CP) algorithm's implementation, comparison is done with the ground truth (GT) values of the corresponding CP estimates. To obtain the GT, an overhead stereo vision system was used which detects the near-exact 2D positions of the robots and the 3D world positions of the soccer ball which the robots in our test-bed (Appendix A) are tracking. Fig. B.1 shows one of the frames of the video footage from the GT system (GTS). The GTS was installed in the Mobile Robotics Lab (LRM - Laboratório de Robótica Móvel) of ISR (Instituto de Sistemas e Robótica), IST (Instituto Superior Técnico), Lisbon, Portugal. Without any significant modifications, except for the stereo baseline length, the GTS is based on a similar setup ([67]) in the LSA lab facility of ISEP Porto[1], Portugal.

B.1 GT System (GTS) Technical Details

Figure B.1: Snapshot from the GTS installed for ground truth evaluation of the ball's 3D positions and the robots' 2D positions.

[1]LSA (Laboratório de Sistemas Autonómos), ISEP http://www.lsa.isep.ipp.pt/

The GTS consists of 2 gigabit ethernet cameras in a stereo baselines. The GTS cameras were positioned looking towards the test-bed with a baseline of $\sim$ 11.11 meters and connected to a machine with Quad Core Intel(R) Core(TM) i5 CPU 750 @ 2.67GHz, 8GB RAM, running a Linux operating system. The cameras used are : Basler acA1300-30gc at $\sim$ 25 frames per second, each of which has a pixel resolution of 1294×964.

Appendix C

Cooperative Extended Kalman Filter

In this appendix, we present the cooperative extended Kalman filter (EKF) formulation for a team of N robots tracking an object $\mathbf{O}$ in an environment with L known and static landmarks. The formulation can be extended to multiple tracked objects with known data association in a straightforward manner.

We denote the robots as $\mathrm{r}_1, .., \mathrm{r}_N$. The state (pose in the world frame) of the robot r_n is given by $\mathcal{L}_t^{\mathrm{r}_n} = [x_t^{\mathrm{r}_n}\ y_t^{\mathrm{r}_n}\ \theta_t^{\mathrm{r}_n}]^\top$.

The state (3D-position and velocity in the world frame) of the tracked moving object is given by $\mathbf{O}_t = [x_t^{\mathbf{o}}\ y_t^{\mathbf{o}}\ z_t^{\mathbf{o}}\ \mathbf{v}_{x_t}^{\mathbf{o}}\ \mathbf{v}_{y_t}^{\mathbf{o}}\ \mathbf{v}_{z_t}^{\mathbf{o}}]^\top$ at the t^{th} timestep.

The 2D-position of the l^{th} known and static landmark is given by $\mathbf{l}^l = [\mathbf{l}_x^l\ \mathbf{l}_y^l]^\top$. The landmarks are assumed to be fixed on the ground plane on which the robots move.

The odometry measurement made by the robot r_n at the t^{th} timestep is given by $\mathbf{u}_t^{\mathrm{r}_n}$, and an associated Gaussian noise with zero mean and covariance matrix $\mathbf{R}_t^n$.

The static landmark observation measurement of the l^{th} landmark made by the robot r_n in its local frame at the t^{th} timestep is given by $\mathbf{z}_t^{\mathrm{r}_n,l}$ and an associated Gaussian noise with zero mean and covariance matrix $\mathbf{Q}_t^{\mathrm{r}_n,l}$.

Similarly, the moving object $\mathbf{O}$'s observation measurement made by the robot r_n in its local frame at the t^{th} timestep is given by $\mathbf{z}_t^{\mathrm{r}_n,\mathbf{o}}$ and the associated Gaussian noise with zero mean and covariance matrix $\mathbf{\Sigma}_t^{\mathrm{r}_n,\mathbf{o}}$.

The motion of the object is modeled using a constant velocity motion model with random acceleration. Let $\mathbf{O}_t = \mathbf{A}\mathbf{O}_{t-1} + \nu$, where $\mathbf{A}$ is the matrix modeling discrete-time constant velocity and ν is a zero mean error with covariance matrix $\mathbf{\Sigma}_t^{\mathbf{o}}$.

We now define $\mathbf{x}_t$ as the full state vector being estimated by stacking all individual states at the t^{th} timestep as follows.

$$\mathbf{x}_t = \left[\mathcal{L}_t^{\mathrm{r}_1\top}\ \ldots\ \mathcal{L}_t^{\mathrm{r}_N\top}\ \mathbf{O}_t{}^\top\right]^\top \tag{C.1}$$

$\mathbf{u}_t$ is obtained by stacking all robots' odometry measurements available at the t^{th} timestep as follows.

$$\mathbf{u}_t = \begin{bmatrix} \mathbf{u}_t^{\mathrm{r}_1\top} & \dots & \mathbf{u}_t^{\mathrm{r}_N\top} \end{bmatrix}^\top \tag{C.2}$$

$\mathbf{z}_t$ is obtained by stacking all the observation measurements available at the t^{th} timestep as follows.

$$\mathbf{z}_t = \begin{bmatrix} \mathbf{z}_t^{\mathrm{r}_1,1\top} & \dots & \mathbf{z}_t^{\mathrm{r}_1,L\top} & \dots & \mathbf{z}_t^{\mathrm{r}_N,1\top} & \dots & \mathbf{z}_t^{\mathrm{r}_N,L\top} & \mathbf{z}_t^{\mathrm{r}_1,\mathbf{o}\top} & \dots & \mathbf{z}_t^{\mathrm{r}_N,\mathbf{o}\top} \end{bmatrix}^\top \tag{C.3}$$

Let $\mathbf{P}_t$ denote the covariance matrix of the estimated state $\mathbf{x}_t$.

EKF Prediction Step

The predicted state $\hat{\mathbf{x}}_t$ at timestep t is obtained from the state $\mathbf{x}_{t-1}$ at timestep $t-1$ as per (C.4). Here, the function f represents the dynamics of the system whose state is being estimated.

$$\begin{aligned} \hat{\mathbf{x}}_t &= f_{t-1}(\mathbf{x}_{t-1}); \\ \hat{\mathbf{P}}_t &= \frac{\partial}{\partial \mathbf{x}_{t-1}} f_{t-1}(\mathbf{x}_{t-1}) \mathbf{P}_t \frac{\partial}{\partial \mathbf{x}_{t-1}} f_{t-1}(\mathbf{x}_{t-1})^T + \mathbf{R}_t \end{aligned} \tag{C.4}$$

where

$$f_{t-1} = \begin{bmatrix} f_{t-1}^{\mathrm{r}_1} \\ \vdots \\ f_{t-1}^{\mathrm{r}_N} \\ \mathbf{AO}_{t-1} \end{bmatrix} = \begin{bmatrix} \mathcal{L}_{t-1}^{\mathrm{r}_1} \oplus \mathbf{u}_t^{\mathrm{r}_1} \\ \vdots \\ \mathcal{L}_{t-1}^{\mathrm{r}_N} \oplus \mathbf{u}_t^{\mathrm{r}_N} \\ \mathbf{AO}_{t-1} \end{bmatrix}, \tag{C.5}$$

$$\frac{\partial}{\partial \mathbf{x}_{t-1}} f_{t-1} = \begin{bmatrix} \frac{\partial}{\partial \mathcal{L}_{t-1}^{\mathrm{r}_1}} f_{t-1}^{\mathrm{r}_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial}{\partial \mathcal{L}_{t-1}^{\mathrm{r}_N}} f_{t-1}^{\mathrm{r}_N} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A} \end{bmatrix} \tag{C.6}$$

and $\mathbf{R}_t$ is a block diagonal matrix with all the respective $\mathbf{R}_t^{\mathrm{r}_n}$ on the diagonal and the last diagonal element being $\mathbf{\Sigma}_t^{\mathbf{o}}$.

EKF Update Step

Here we assume that at each frame every robot observes all the landmarks and moving object. In case this is not true, we simply remove the corresponding line in the next equations.

We first compute the innovation error $\mathbf{e}_t$

$$\mathbf{e}_t = \mathbf{z}_t - \hat{\mathbf{z}}(\mathbf{x}_t) \tag{C.7}$$

where $\mathbf{z}_t$ is the measurement vector as defined in (C.3) and $\hat{\mathbf{z}}(\mathbf{x}_t)$ defined as follows

$$\hat{\mathbf{z}}(\mathbf{x}_t) = \begin{bmatrix} \hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_1}, \mathbf{l}^1) \\ \vdots \\ \hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_N}, \mathbf{l}^1) \\ \vdots \\ \hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_1}, \mathbf{l}^L) \\ \vdots \\ \hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_N}, \mathbf{l}^L) \\ \hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_1}, \mathbf{O}_t) \\ \vdots \\ \hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_N}, \mathbf{O}_t) \end{bmatrix} \tag{C.8}$$

is the prediction of all the measurements in $\mathbf{z}_t$.

We then compute the innovation covariance $\mathbf{S}_t$

$$\mathbf{S}_t = \frac{\partial}{\partial \mathbf{x}_t} \hat{\mathbf{z}}(\mathbf{x}_t) \hat{\mathbf{P}}_t \frac{\partial}{\partial \mathbf{x}_t} \hat{\mathbf{z}}(\mathbf{x}_t)^T + \mathbf{Q}_t \tag{C.9}$$

where $\frac{\partial}{\partial \mathbf{x}_t}\hat{\mathbf{z}}(\mathbf{x}_t) =$

$$\begin{bmatrix}
\frac{\partial}{\partial \mathcal{L}_t^{\mathrm{r}_1}}\hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_1}, \mathbf{l}^1) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \frac{\partial}{\partial \mathcal{L}_t^{\mathrm{r}_N}}\hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_N}, \mathbf{l}^1) & \mathbf{0} \\
\vdots & \vdots & \vdots & \vdots \\
\frac{\partial}{\partial \mathcal{L}_t^{\mathrm{r}_1}}\hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_1}, \mathbf{l}^L) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \frac{\partial}{\partial \mathcal{L}_t^{\mathrm{r}_N}}\hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_N}, \mathbf{l}^L) & \mathbf{0} \\
\frac{\partial}{\partial \mathcal{L}_t^{\mathrm{r}_1}}\hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_1}, \mathbf{O}_t) & \mathbf{0} & \mathbf{0} & \frac{\partial}{\partial \mathbf{O}_t}\hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_1}, \mathbf{O}_t) \\
\mathbf{0} & \ddots & \mathbf{0} & \vdots \\
\mathbf{0} & \mathbf{0} & \frac{\partial}{\partial \mathcal{L}_t^{\mathrm{r}_N}}\hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_N}, \mathbf{O}_t) & \frac{\partial}{\partial \mathbf{O}_t}\hat{\mathbf{z}}(\mathcal{L}_t^{\mathrm{r}_N}, \mathbf{O}_t)
\end{bmatrix} \tag{C.10}$$

and $\mathbf{Q}_t$ is a block diagonal matrix with all the respective $\mathbf{Q}_t^{\mathrm{r}_n}$ and $\mathbf{\Sigma}_t^{\mathrm{r}_n,\mathbf{o}}$ on the diagonal.

Given the innovation, we compute the Kalman gain as follows

$$\mathbf{K}_t = \hat{\mathbf{P}}_t \frac{\partial}{\partial \mathbf{x}_t}\hat{\mathbf{z}}(\mathbf{x}_t)^T \mathbf{S}_t^{-1} \tag{C.11}$$

and update the filter state using the following equations

$$\mathbf{x}_t = \hat{\mathbf{x}}_t + \mathbf{K}_t \mathbf{e}_t \tag{C.12}$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t \frac{\partial}{\partial \mathbf{x}_t}\hat{\mathbf{z}}(\mathbf{x}_t))\hat{\mathbf{P}}_t \tag{C.13}$$

References

[1] T. Bailey, M. Bryson, H. Mu, J. Vial, L. McCalman, and H. Durrant-Whyte, "Decentralised cooperative localisation for heterogeneous teams of mobile robots," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 2859 –2865.

[2] K. Zhou and S. Roumeliotis, "Multirobot active target tracking with combinations of relative observations," *Robotics, IEEE Transactions on*, vol. 27, no. 4, pp. 678 –695, aug. 2011.

[3] A. I. Mourikis and S. I. Roumeliotis, "Predicting the performance of cooperative simultaneous localization and mapping (c-slam)," *Int. J. Rob. Res.*, vol. 25, no. 12, pp. 1273–1286, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1177/0278364906072515

[4] L.-L. Ong, B. Upcroft, T. Bailey, M. Ridley, S. Sukkarieh, and H. Durrant-Whyte, "A decentralised particle filtering algorithm for multi-target tracking across multiple flight vehicles," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, oct. 2006, pp. 4539 –4544.

[5] L.-L. Ong, B. Upcroft, M. Ridley, T. Bailey, S. Sukkarieh, and H. Durrant-Whyte, "Consistent methods for decentralised data fusion using particle filters," in *Multisensor Fusion and Integration for Intelligent Systems, 2006 IEEE International Conference on*, sept. 2006, pp. 85 –91.

[6] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with gmm approximation for multiple targets localization and tracking in wireless sensor network," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, april 2005, pp. 181 – 188.

[7] W. Nisticò, M. Hebbel, T. Kerkhof, and C. Zarges, "Cooperative visual tracking in a team of autonomous mobile robots," *RoboCup 2006: Robot Soccer World Cup*

X. Lecture Notes In Artificial Intelligence, pp. 146–157, 2007. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74024-7_13

[8] A. Ahmad, G. D. Tipaldi, P. Lima, and W. Burgard, "Cooperative robot localization and target tracking based on least square minimization," in *Robotics and Automation, 2013. ICRA 2013. Proceedings of the 2013 IEEE International Conference on*, May 2013.

[9] A. Ahmad and P. Lima, "Multi-robot cooperative object tracking based on particle filters," in *Proc. of the European Conference on Mobile Robots (ECMR 2011)*, Örebro, Sweden, Sep 2011.

[10] A. Ahmad and P. Lima, "Multi-robot cooperative spherical-object tracking in 3d space based on particle filters," *Accepted in the Robotics and Autonomous Systems journal, Special Issue on ECMR'11 (To appear in 2013).*

[11] A. Ahmad, T. Nascimento, A. G. S. Conceio, A. P. Moreira, and P. Lima, "Perception-driven multi-robot formation control," in *Robotics and Automation, 2013. ICRA 2013. Proceedings of the 2013 IEEE International Conference on*, May 2013.

[12] P. U. Lima, P. Santos, R. Oliveira, A. Ahmad, and J. Santos, "Cooperative localization based on visually shared objects," *RoboCup 2010: Robot Soccer World Cup XIV. Lecture Notes In Artificial Intelligence*, pp. 350–361, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1983806.1983837

[13] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006. [Online]. Available: http://doi.acm.org/10.1145/1177352.1177355

[14] C. Kwok and D. Fox, "Map-based multiple model tracking of a moving object," *RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes In Artificial Intelligence*, pp. 18–33, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=2168172.2168176

[15] P. A. V. Raúl A. Lastra and J. R. del Solar, "Integrated self-localization and ball tracking in the four-legged robot soccer league," in *1st IEEE Latin American Robotics Symposium*, Mexico City, Mexico, 2004.

[16] J. Inoue, A. Ishino, and A. Shinohara, "Ball tracking with velocity based on monte-carlo localization." in *IAS*, T. Arai, R. Pfeifer, T. R. Balch,

and H. Yokoi, Eds. IOS Press, 2006, pp. 686–693. [Online]. Available: http://dblp.uni-trier.de/db/conf/ias/ias2006.html#InoueIS06

[17] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 9, pp. 1820 –1833, sept. 2011.

[18] D. Gohring and H.-D. Burkhard, "Multi robot object tracking and self localization using visual percept relations," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, oct. 2006, pp. 31 –36.

[19] L. Merino, F. Caballero, J. Martnez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple uavs: Application to automatic detection of forest fires," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 165–184, 2006. [Online]. Available: http://dx.doi.org/10.1002/rob.20108

[20] M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino, and P. Lima, "Tracking objects with generic calibrated sensors: An algorithm based on color and 3d shape features," *Robotics and Autonomous Systems*, vol. 58, no. 6, pp. 784 – 795, 2010, omnidirectional Robot Vision. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889010000552

[21] J. Santos and P. U. Lima, "Multi-robot cooperative object localization: decentralized bayesian approach," *RoboCup 2009: Robot Soccer World Cup XIII. Lecture Notes In Artificial Intelligence*, pp. 332–343, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=2167873.2167902

[22] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, pp. 325–344, 2000.

[23] A. Howard, M. Matarić, and G. Sukhatme, "Localization for mobile robot teams: A distributed mle approach," in *Experimental Robotics VIII*, ser. Springer Tracts in Advanced Robotics, B. Siciliano and P. Dario, Eds. Springer Berlin Heidelberg, 2003, vol. 5, pp. 146–155.

[24] K. Leung, T. Barfoot, and H. Liu, "Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach," *Robotics, IEEE Transactions on*, vol. 26, no. 1, pp. 62 –77, feb. 2010.

[25] R. Madhavan, K. Fregene, and L. E. Parker, "Distributed heterogeneous outdoor multi-robot localization," in *In Proceedings of the IEEE International Conference on Robotics and Automation*, 2002, pp. 374–381.

[26] H. Mu, T. Bailey, P. Thompson, and H. Durrant-Whyte, "Decentralised solutions to the cooperative multi-platform navigation problem," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 47, no. 2, pp. 1433 –1449, april 2011.

[27] A. C. Sanderson, "A distributed algorithm for cooperative navigation among multiple mobile robots," *Advanced Robotics*, vol. 12, no. 4, pp. 335–349, 1997. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1163/156855398X00235

[28] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A Probabilistic Approach to Collaborative Multi-Robot Localization," *Autonomous Robots*, vol. 8, no. 3, 2000.

[29] S. I. Roumeliotis and G. Bekey, "Distributed Multirobot Localization," *IEEE Transactions on Robotics*, vol. 18, no. 5, pp. 781–795, Oct 2002.

[30] X. S. Zhou and S. I. Roumeliotis, "Robot-to-Robot Relative Pose Estimation from Range Measurements," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1168–1185, 2008.

[31] J. W. Fenwick, P. M. Newman, and J. J. Leonard, "Cooperative Concurrent Mapping and Localization," in *Proc. of the IEEE Intl. Conf. on Rob. and Autom.*, 2002.

[32] C. Jennings, D. Murray, and J. Little, "Cooperative Robot Localization with Vision-Based Mapping," in *Proc. of the IEEE Intl. Conf. on Rob. and Autom.*, vol. 4, 1999, pp. 2659–2665.

[33] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[34] A. J. Neves, A. J. Pinho, D. A. Martins, and B. Cunha, "An efficient omnidirectional vision system for soccer robots: From calibration to object detection," *Mechatronics*, vol. 21, no. 2, pp. 399 – 410, 2011, special Issue on Advances in intelligent robot design for the Robocup Middle Size League. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957415810000863

[35] D. Schneider, E. Schwalbe, and H.-G. Maas, "Validation of geometric models for fisheye lenses," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 3, pp. 259 – 266, 2009, theme Issue: Image Analysis and Image Engineering in Close Range Photogrammetry. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271609000045

[36] A. J. P. Daniel A. Martins, Antnio J. R. Neves, "Real-time generic ball detection in robocup domain," in *IBERAMIA'08 - Ibero-American Conference on Artificial Intelligence: IROBOT'08 - 3rd International Workshop on Intelligent Robotics, Lisboa, Portugal*, October 2008, pp. 37–48.

[37] A. Voigtlnder, S. Lange, M. Lauer, and M. A. Riedmiller, "Real-time 3d ball recognition using perspective and catadioptric cameras." in *European Conference on Mobile Robotics (EMCR), Freiburg, Germany*, 2007.

[38] A. Neves, J. L. Azevedo, B. Cunha, J. Cunha, R. Dias, P. Fonseca, N. Lau, E. Pedrosa, A. Pereira, and J. Silva, "Cambada'12: Team description paper," in *RoboCup 2012, Mexico City, Mexico*, 2012.

[39] M. Wenig, K. Pang, and P. On, "Arbitrarily colored ball detection using the structure tensor technique," *Mechatronics*, vol. 21, no. 2, pp. 367 – 372, 2011, special Issue on Advances in intelligent robot design for the Robocup Middle Size League. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957415810001297

[40] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972. [Online]. Available: http://doi.acm.org/10.1145/361237.361242

[41] S. Suzuki and K. be, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32 – 46, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0734189X85900167

[42] J. Sklansky, "Finding the convex hull of a simple polygon," *Pattern Recognition Letters*, vol. 1, no. 2, pp. 79 – 83, 1982. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167865582900162

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

[44] J. S. Liu, "Metropolized independent sampling with comparisons to rejection sampling and importance sampling," *Statistics and Computing*, vol. 6, pp. 113–119, Jun 1996, 10.1007/BF00162521. [Online]. Available: http://dx.doi.org/10.1007/BF00162521

[45] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, april 2005, pp. 2432 – 2437.

[46] S. Lenser and M. Veloso, "Sensor Resetting Localization for Poorly Modelled Mobile Robots," in *Proc. of the IEEE Intl. Conf. on Rob. and Autom.*, 2000, san Francisco, CA, USA.

[47] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for Mobile Robots," *Artificial Intelligence*, vol. 128, no. 1–2, pp. 99–141, May 2001.

[48] J. Rogers, A. Trevor, C. Nieto-Granda, and H. Christensen, "Slam with expectation maximization for moveable object tracking," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, oct. 2010, pp. 2077 –2082.

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

[50] L.-L. Ong, B. Upcroft, T. Bailey, M. Ridley, S. Sukkarieh, and H. Durrant-Whyte, "A decentralised particle filtering algorithm for multi-target tracking across multiple flight vehicles," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, oct. 2006, pp. 4539 –4544.

[51] L.-L. Ong, T. Bailey, H. Durrant-Whyte, and B. Upcroft, "Decentralised particle filtering for multiple target tracking in wireless sensor networks," in *Information Fusion, 2008 11th International Conference on*, 30 2008-july 3 2008, pp. 1 –8.

[52] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, may 2010, pp. 3185 –3192.

[53] H. Wang, G. Hu, S. Huang, and G. Dissanayake, "On the structure of nonlinearities in pose graph SLAM," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[54] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Transactions on Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 2010.

[55] W. Blair, "Design of nearly constant velocity track filters for tracking maneuvering targets," in *Information Fusion, 2008 11th International Conference on*, 30 2008-july 3 2008, pp. 1 –7.

[56] P. Quang, C. Musso, and F. Le Gland, "An insight into the issue of dimensionality in particle filtering," in *Information Fusion (FUSION), 2010 13th Conference on*, july 2010, pp. 1 –8.

[57] F. Daum and J. Huang, "Curse of dimensionality and particle filters," in *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, vol. 4, 8-15, 2003, pp. 4_1979 – 4_1993.

[58] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners," *Signal Processing, IEEE Transactions on*, vol. 50, no. 3, pp. 736 –746, mar 2002.

[59] F. Le Gland and N. Oudjane, "Stability and Uniform Approximation of Nonlinear Filters Using the Hilbert Metric and Application to Particle Filters," *The Annals of Applied Probability*, vol. 14, no. 1, pp. 144–187, 2004. [Online]. Available: http://dx.doi.org/10.2307/4140493

[60] S. Thrun, "Particle filters in robotics," in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, ser. UAI'02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 511–518. [Online]. Available: http://dl.acm.org/citation.cfm?id=2073876.2073937

[61] C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson, "Obstacles to High-Dimensional Particle Filtering," *Mon. Wea. Rev.*, vol. 136, no. 12, pp. 4629–4640, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1175/2008MWR2529.1

[62] F. Zhang and N. Leonard, "Cooperative filters and control for cooperative exploration," *Automatic Control, IEEE Transactions on*, vol. 55, no. 3, pp. 650 –663, march 2010.

[63] T. P. Nascimento, F. A. Fontes, A. P. Moreira, and A. G. S. Conceio, "Leader following formation control for omnidirectional mobile robots - the target chasing problem."

in *ICINCO (2)*, J.-L. Ferrier, A. Bernard, O. Y. Gusikhin, and K. Madani, Eds. SciTePress, 2011, pp. 135–144.

[64] T. P. Nascimento, "Coordinated Multi-Robot Formation Control," PhD, Porto University, 2012.

[65] L. Oliveira, L. Almeida, and F. Santos, "A loose synchronisation protocol for managing rf ranging in mobile ad-hoc networks," in *RoboCup 2011: Robot Soccer World Cup XV*, ser. Lecture Notes in Computer Science, T. Rfer, N. Mayer, J. Savage, and U. Saranl, Eds. Springer Berlin Heidelberg, 2012, vol. 7416, pp. 574–585. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32060-6_49

[66] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainly," *Int. J. Rob. Res.*, vol. 5, no. 4, pp. 56–68, Dec. 1986. [Online]. Available: http://dx.doi.org/10.1177/027836498600500404

[67] H. Silva, A. Dias, J. Almeida, A. Martins, and E. Silva, "Real-time 3d ball trajectory estimation for robocup middle size league using a single camera," *RoboCup 2011: Robot Soccer World Cup XV. Lecture Notes In Artificial Intelligence*, vol. 7416, pp. 586–597, 2012. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-32060-6_50