INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Multi-Robot Cooperative Object Localization

Decentralized Bayesian Approach

João Carlos da Silva Santos

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica e de Computadores

Júri

Presidente:	Carlos J. Silvestre
Orientador:	Pedro U. Lima
Vogais:	Alexandre M. Bernardino

Dezembro de 2008

Acknowledgments

Aqui, agradeço a todos aqueles que pelos mais variados motivos considero que tiveram uma enorme influência no meu trabalho, na minha formação e no fundo contribuíram para quem eu sou hoje...

À minha família, por um apoio sem igual, pela dedicação, esforço e pela força de acreditar. Obrigada Mãe, obrigado Pai...

A todos os meus amigos... Isto teria sido bem mais complicado e muito menos divertido sem vós!

Obrigado a todos aqueles que comigo fizeram parte da SCDEEC, foi um prazer fazer parte desta equipa.

Obrigado a todos os BESTies, foi uma aventura inesquecível e um lema que ficará para sempre: "work hard, party harder"!.

Obrigado também a todos aqueles que contribuíram para o projecto ISocRob, por aquilo que me possibilitaram aprender convosco, por todas as experiências e pela vivência dos torneios. Vou sentir falta desta adrenalina... e vou ter saudades do omni5!

À ENTeam: Estilita, Marco e Nelson, isto tem sido uma viagem.... Obrigado por tudo aquilo que passámos juntos... E agora que venham mais sucessos ;)

Ao Prof. Pedro Lima, obrigado, não ao orientador (apesar das dores de cabeça que lhe causei), não ao professor (apesar dos bons ensinamentos), mas obrigado essencialmente ao apaixonado pela robótica que acolheu no ISR, da melhor maneira possível, 4 caloiros com muita vontade de brincar aos robôs... Obrigado por tudo aquilo que me proporcionou, e por ter marcado os anos que passei no Técnico da melhor forma.

Por muito mais do que aquilo que eu te posso algum dia agradecer, obrigado pela paciência, pelo apoio e pela inspiração.... Obrigado Sofia.

Abstract

When operating in a complex unstructured environment, a team of cooperative robots becomes a team of sensors, each making observations to build a perception of reality that can be improved by others. A sensor model describes the uncertainty associated with each observation allowing to extract relevant information, rather than simple raw data from a physical device.

The sensor models are often nonlinear resulting in non-Gaussian posterior distributions. However, a parametric (e.g. Gaussian) approximation of sensors information is usually a better choice given the low computational power and low communications bandwidth it requires when sharing information. This is achieved at the cost of a limited representation of the sensors belief. Non parametric discrete approximations, such as Particle Filters, are able to capture arbitrarily complex uncertainty, but are intractable when it comes to communicating the state distribution due to the necessity of transmitting a large sample-based representation.

We aim at developing a cooperative sensor fusion model for mobile robots acting in dynamic environments. Our case study is the RoboCup MSL, where we implemented a shape-based 3D tracker for the target at hand: the ball. Furthermore, we aim at conceiving a more accurate probabilistic representation of the information shared between sensors, that copes with nonlinear sensor models. We took particle filters, Gaussian Mixture Model and a decentralized Bayesian approach to propose a cooperative sensor model that improves ball tracking and self-localization.

Keywords

Decentralized Sensor Fusion, Distributed Particle Filter, Gaussian Mixture Models, 3D Tracking, RoboCup

Resumo

Em ambientes pouco estruturados, uma equipa de robôs torna-se numa equipa de sensores, cada qual fazendo observacões para construir uma percepção da realidade que pode ser melhorada por outros. Um modelo do sensor descreve a incerteza associada a essas observações, permitindo recolher informações mais relevantes do que leituras de dados de um dispositivo.

Os modelos de sensores tipicamente são não lineares, originando distribuições não Gaussianas sobre a certeza das observações efectuadas. Aproximar a informação do sensor a uma Gaussiana é normalmente uma boa opção devido à sua parametrização simples, que não leva a uma estrutura de dados complexa que ocupe muita largura de banda quando se partilha informação. No entanto, quando se incorre em tal aproximação a representação da crença de um sensor é limitada. Outros tipos de aproximações não lineares, como Filtros de Partículas, são capazes de captar melhor a incerteza de um sensor, mas não é suportável do ponto de vista da comunicação transmitir todas as partículas que aproximam essa incerteza.

Foi desenvolvido um modelo de percepção cooperativa para robôs móveis que operam em ambientes dinâmicos. O modelo foi aplicado a robôs futebolistas, onde primeiro implement/'amos um seguidor 3D capaz de seguir uma bola pela sua forma. Iremos ainda conceber um método de representação probabilística para a informação partilhada entre sensores, que lida com modelos não lineares. Apresentamos no fim a nossa abordagem descentralizada, com base em filtros de partículas, modelos de misturas Gaussianas e no filtro de Bayes, para implementar um modelo de percepção cooperativa capaz de melhorar a estimativa da bola e a auto-localização.

Palavras Chave

Fusão Sensorial Descentralizada, Filtro de Partículas Distribuido, Modelos de Misturas de Gaussianas, Tracking em 3D, RoboCup

Contents

List of Figures

List of Tables

Acronyms

CI	Covariance Intersection
DDF	Decentralized Data Fusion
EPM	Equidistance Projection Model
EM	Expectation Maximization
GMM	Gaussian Mixture Model
IPP	Intel Performance Primitives
ISocRob	Intelligent Soccer Robots
ISR	Institute for Systems and Robotics
IST	Instituto Superior Técnico
MeRMaID	Multiple-Robot Middleware for Intelligent Decision-making
MCL	Monte Carlo Localization
MKL	Math Kernel Library
MSL	Middle Size League
pdf	Probability Density Function
PF	Particle Filter

1

Introduction

Contents

The need for knowledge concerning the external world has always been a fundamental issue in the history of mankind. Human beings have a special mechanism for acquiring this knowledge, known as sense perception. Autonomous systems must be able to carry out a similar task in obtaining an internal description of the external environment. This is obtained from the use of sensor devices that generate measurements as a function of received signals or stimulus. However, a single sensor does not always provide to the system all the required information that enables it to update its own model of the world in a reliable way. Naturally, the need for multiple sensors derives from the need of providing a better and more precise understanding of the environment. Multisensor Fusion addresses the problem of combining all the information from multiple sensors in order to yield a consistent and coherent description of the observed environment. The problem itself comes from the fact that the sensors information is always uncertain, usually partial, occasionally incorrect and often geographically or geometrically incomparable with other sensor views.

Multisensor systems have been applied to several areas such as aerospace, military, automated manufacturing, process control, power generations and robotics. This work focus on robotics, where the word ”sensor” has nowadays a broader sense. When operating in a complex unstructured environment, a team of cooperative robots becomes in fact a team of sensors, each making observations to build a perception of reality that can be improved by others. A sensor model describes the uncertainty associated with each observation allowing to extract relevant information, rather then simple raw data from a physical device. This can be achieved by both probabilistic or non-probabilistic techniques. We take the probabilistic approach and represent the sensor model as a probability density function (pdf).

Robots act as mobile sensors that are part of a network where each node communicates with other by means of a wireless communication medium. This can either be a common communication facility with broadcast or a point-to-point architecture. Sensor fusion can be concentrated in one single node or distributed across all the nodes. The best network topology must be chosen according to the application at hand.

1.1 Motivation

Although it aims at modeling generic mobile multisensor systems, the following work focus on the specific application to soccer robotics. It was developed under the ISocRob project [1] for omnidirectional robots playing in the RoboCup Middle Size League (MSL). Soccer robots are equipped with an omnidirectional camera with limited resolution that hardly provides a global view of the field. Our main motivation is to take real advantage of this team of mobile sensors scattered across the field, in order to provide a broader view while locating and tracking the ball. We are further motivated in benefiting from a multisensor system upon the challenges constantly imposed

by RoboCup such as the global localization in a symmetric environment or the tracking of the (yet to come) arbitrary color ball.

1.2 Objectives

We aim at developing a cooperative sensor fusion model for mobile robots acting in dynamic environments. Our case study is the RoboCup MSL, where we will implement a shape-based 3D tracker for the target at hands: the ball. Furthermore, we aim at conceiving a more accurate probabilistic representation of the information shared between sensors, that copes with nonlinear sensor models. We will then take particle filters, Gaussian Mixture Model and a decentralized Bayesian approach to propose a cooperative sensor model that improves ball tracking and self-localization.

1.3 Main contributions

Our main contributions in the following work are:

- the extension and application to mobile robots of the previous work of Taiana [2] on 3D model-based tracking with particle filters;
- a framework for representing and measuring disagreement of sensor information based on Gaussian Mixture Models;
- a cooperative sensor fusion model based on a particle filter perception framework.

1.4 Dissertation outline

In Section 2 we review the current state of the art concerning three major topics addressed in this thesis and its evolution in RoboCup: 3D visual tracking for moving objects, distributed sensor networks and data fusion techniques used to combine multiple information. In Section 3 we describe the two components of our sensor fusion model: the self-localization method already in use in the ISocRob omnidirectional robots, and the implementation of the ball detection and tracking algorithm. Then in Section 4 we present a compact sensor information representation based on GMMs and introduce a decentralized Bayesian approach to multisensor fusion that takes advantage of distributed particle filters and GMM modeling. In Section 5 we describe our experimental setup and present several experimental results to validate the introduced methods. Section 6 outlines our conclusions and future work.

2

Background and Related Work

Contents

2.1 3D Visual Tracking

Up until recently, all elements of a RoboCup game had distinct colors, such as blue and yellow goals and an orange ball, and so most detection and tracking methods were based on color domain approaches. Furthermore, the tracking problem was made in 2D, for there was no need to go further since the ball was always played at ground level. Over the years, RoboCup as always pushed the rules one step further in order to increase the complexity of the challenge. New rules made it hard for color domain approaches to succeed with the inclusion of natural light over the field and uncolored goals. Although extremely fast, object detection algorithms relying strictly on color classification such as [3], using solid look up tables for image labeling, were no longer accurate nor reliable. At this point, the object shape was introduced as a plausibility measure such as the one implemented in [4] by a spline calibration curve which points out the pixel size of the ball at different distances.

Moreover, the introduction of new powerful kickers allowed the robots to kick the ball way above ground floor, making it to bounce all over the field. With a stereo vision system composed of one omnidirectional and one perspective camera Hafner et al. [5] determined the ball three dimensional position by geometric reasoning, finding the closest point to two skewed lines which are defined from the two camera images. However, stereo vision systems require for the object to be found in both camera images, and even tho the robot is always trying to face it, typically, a ball above 1 meter cannot be seen on the omnidirectional camera image.

Objects motion follows the laws of physics, so defining an appropriate dynamic model for the target is crucial. Prior work for the calculation of the ball movement used a Kalman filter to estimate both position and velocity. This is based on the assumption of linear ball movement with constant velocity, making it hard to track the ball when sudden changes in the movement occurs, i.e., kicking our bouncing. Instead, Lauer et al. [6] proposed the use of a direct approach that recalculates the motion parameters every iteration directly from the latest observations. Also to address this issue, a very wide spread approach is the use of particle filtering. Taiana [2] implemented a system based on particle filter, comprising a 3D shape model and a 3D motion model, using either an omnidirectional or a perspective camera.

2.2 Distributed Sensor Networks

The problem of detecting and tracking moving objects in distributed sensor networks has been widely studied. Such networks comprise a multi-sensor system employing several sensors to obtain information about a real world environment full of uncertainties. Each sensor is part of a network node which has local computational power and is able to communicate with nearby sensors. Different network architectures correspond to different information fusion algorithms that can be reduced to three general categories: centralized, hierarchical and decentralized. In cen-

tralized architectures, data fusion computational load and communications are carried out by a single central processor. Therefore it is brittle because it possesses a single point of failure. Hierarchical architectures ease the computational load by aggregating sensors information in local fusion centers, and then passing the local fused data to the central fusion processor. Yet, global information depends on one single unit. These drawbacks are solved by using decentralized architectures, where all fusion processes take place locally and no global knowledge of the network is required *a priori*. However, communications overheads are higher than in other topologies. Detailed description, advantages and limitations of these architectures can be found in [7], [8].

In the RoboCup domain, first approaches on distributed sensor fusion were based on centralized architectures. Dietl et al. [9] [10] implementation to globally perceive the ball and other players position was based on fusing robots observation in a central computer outside the soccer field. In fact, most data fusion algorithms have been developed for centralized topologies because it's considered an optimal approach if one discards communication problems and has enough computational resources [11]. On the other hand, one can only take real advantage of hierarchical architectures on large scale systems, such as the rescue simulation league scenario, where it's possible to create several fusion layers [11]. New rules to break communications with exterior elements now impose for decentralized topologies or centralized based topologies with a dynamic leader node [12]. Several teams have taken the decentralized way for a fully multi-agent approach [3] [4] [5] [13]. However, the implementations described rely mostly on parametric sensor models which do not overload communications when passing information on sensor belief or observations. We propose a decentralized approach based on a probabilistic framework from non-parametric sensors, where communication constraints must be taken into account.

2.3 Data Fusion Techniques

Most of the previous work on RoboCup focus on merging the ball to one consistent estimate among the team of robots. Lau et al. [14] simply calculate the mean and standard deviation of all ball estimates for discarding outliers and then assumes the ball information of the teammate closest to it. Ferrein et al. [15] describe a weighted mean of the estimates according to the distance from the robot to the ball and a time factor denoting how long ago the robot has seen the ball for the last time. On a more probabilistic approach, Stroupe et al. [16] represent ball estimates as a two-dimensional gaussian in canonical form, allowing to merge them by multiplication, and use a Kalman filter to predict the ball position. Pinheiro and Lima [17] implemented a multi-Bayesian team of robots as a direct application of the sensor fusion method introduced by Durrant-Whyte [7]. This allowed for team members to achieve more frequent cooperation, so as to detect sensors disagreement based on the Mahalanobis distance and achieve a team consensus faster. Other approaches also accounted for merging weighted gridcells from ball occupancy maps [18],

Monte Carlo (ball) localization [19] or a combination of Kalman filter with Markov localization [9]. However, although mentioned in some approaches, none of these take real consideration in the robots own localization estimate, frequently assuming a high accuracy self-localization method.

Pahliani and Lima [20] introduced a new cooperative localization algorithm that reduces the uncertainty of both self-localization and object localization. This method tries to overcome the performance of two popular algorithms for fusing sensor observations: Linear Opinion Pool and Logarithmic Opinion Pool. The implementation although, is based on multi-robot Markov Localization and assumes one can distinguish and locate different team-mates, which is a complex task given the current RoboCup environment.

On other domains, Rosencrantz, Gordon and Thrum [21] proposed a scalable Bayesian technique for decentralized state estimation with distributed particle filters using a selective communication procedure over the particle set. On the other hand, instead of selecting which particles to communicate, Durrant-Whyte et al. [22] demonstrated the validity of approximating a particle set using Gaussian mixture models or Parzen representations in DDF systems.

3

Probabilistic World Perception

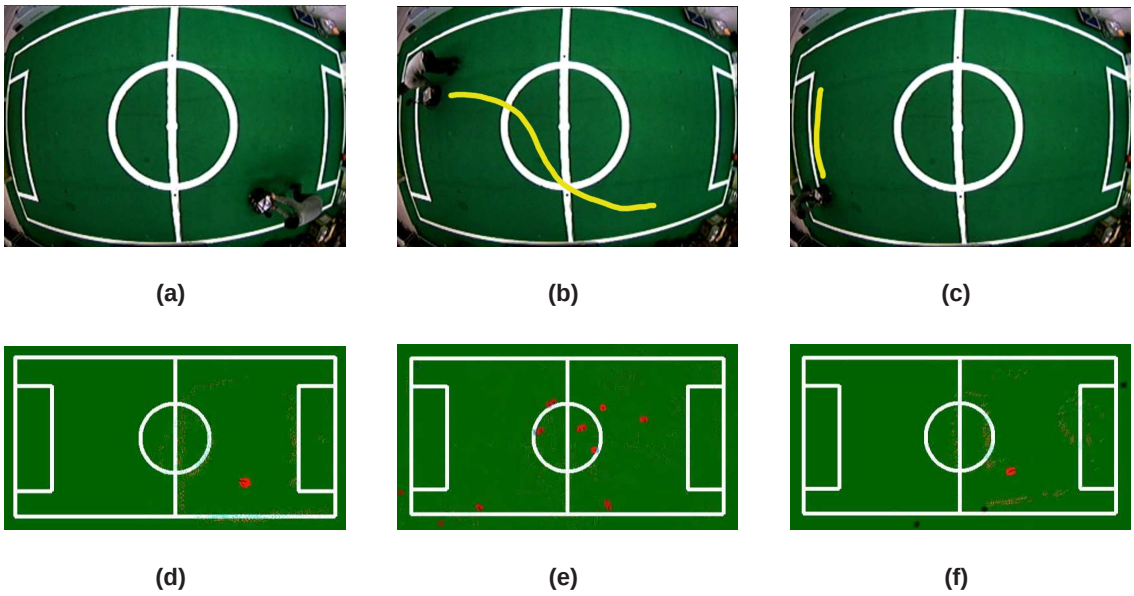
Contents

3.1 Self-Localization

The current self-localization method is based on the previous work of Messias, Santos, Estilita and Lima [23], in which we combine Monte Carlo Localization with gyrodometry and line points extraction. MCL has rapidly become one of the most popular approaches to the localization problem since it's introduction by Fox in 1999 [24]. The algorithm consists in the implementation of the particle filter applied to robot localization, representing the belief $bel(l_t)$ by a set of M weighted samples or particles $\mathbf{R}_t = \{\mathbf{r}_t^{[1]}, \mathbf{r}_t^{[2]}, ..., \mathbf{r}_t^{[M]}\}$. MCL derives from the basic Markov Localization algorithm therefore, the belief $bel(l_t)$ over all possible postures is obtained recursively over $bel(l_{t-1})$ every time a new movement is performed (*prediction*) and a new sensor measurement is taken (*update*). The particle set constitutes a discrete approximation of the belief function where each sample $\mathbf{r}_t$ contains a posture information (x, y, θ) and a numerical weighting factor w. When the robot moves, new samples are generated, each by randomly drawing from the the previous computed particle set with likelihood determined by w. Each new particle posture is then updated according to the motion model. For a new sensor measurement, all sample weights are updated according to the sensor model. With MCL we can approximate the posterior to any distribution of practical importance. This means that we are not bounded to a parametric subset of distributions, such as in the localization approaches based on Kalman Filter.

Some improvements have been made since the first implementation in [23]. A method for iteratively improving the pose estimation based on forces exerted by the model lines over the transformed line points, as described in [25], was implemented. For this work implementation we had to ease computational load by modifying the line points extraction algorithm to process smaller images with half the resolution. Also, to prevent the computation of delayed sensor data we establish a simple synchronization protocol that discards outdated sensor information. This improvements allowed for a faster and more accurate localization method, keeping track of the robot posture with only 20 particles.

One of the issues that affects MCL performance is the ability to recover from failures. When a robot believes it is localized it means that only the particles near the most likely posture have "survived". If the most likely posture happens to be incorrect, the algorithm must be able to recover from it, otherwise the robot won't localize itself anymore. This is often referred in the literature as the kidnapping problem and is the hardest problem to solve in global localization [26]. Our approach deals with kidnapping by adding a variable number of uniformly distributed particles to the sample set, that depends directly on the mismatch between sensor measurements and sensor model. However, during RoboCup game situations this proved to be a risk solution. Due to sensor noise and the dynamic environment, sensor model mismatch has a large fluctuation which leads the algorithm to sometimes "jeopardize" the localization by assuming false kidnapping situations, as shown in fig. 3.1. To prevent this, a new approach to deal with global localization failures will



(a) (b) (c)

(d) (e) (f)

Figure 3.1: A false kidnapping situation occurs due to sensor noise while the robot is being pushed. In the end it led to an incorrect posture by the MCL algorithm. (a) Real starting location. (b) Location at the time false kidnapping was detected. (c) Real end point. (d) All particles convey to the same posture. (e) Particles trying to recover from kidnapping. (f) Final posture estimate.

be proposed further in 4.2.

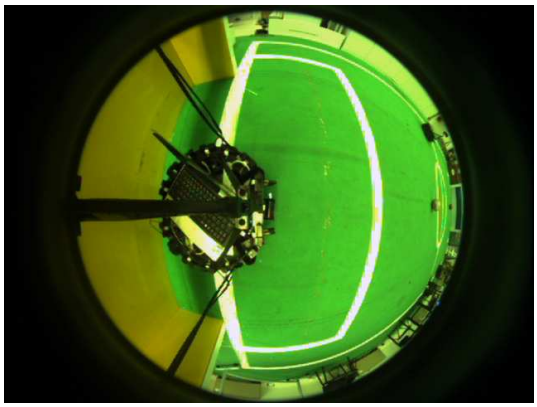
3.2 Ball Detection and Tracking

3.2.1 3D Projection Model

ISocRob robots have a dioptric vision system with a camera facing downwards. The camera has a fish-eye lens providing a field-of-view of 185°, and therefore gives the robot an omnidirectional view in the azimuth direction, as observed in fig. 3.2(a). Although providing the capability to extract information from a 360° view of the environment, the image resolution is much higher near the robot, making it impossible to perform a reliable detection of a ball at more than 5 meters away. Given the current official field size ($18m \times 12m$), one should be able to detect more distant objects. To do so, we have installed a new camera facing forward also coupled with a fish-eye lens. On front, we aim at having a wide angle of view along the horizontal azimuth direction angle, therefore using a camera with a smaller CCD sensor format ($1/3''$) we get a field-of-view of 160°, see fig. 3.2(b).

Most common fish-eye lens are designed to obey the Equidistance Projection Model, described as:

$$r = f\theta \tag{3.1}$$



(a)

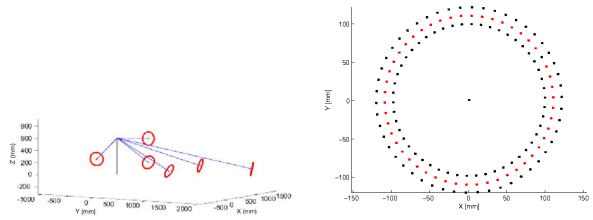
(b)

Figure 3.2: Vision system. (a) Omnidirectional 185° view image. (b) Front 160° wide view image.

where θ is the angle between the principal axis and the incoming ray, r is the distance between the image point and the principal point and f is the focal length.

The ball identification in the image is based on Taiana [2] ball projection model. A 3D model of the ball is used to calculate its 2D contour projected on the image. The ball has rotational symmetry which reduces the problem dimension for there is no need to consider the object orientation. If one considers the polygonal model of a sphere, the ball contour on the image plane lies on the intersection with an orthogonal plane to the line connecting the projection center to the center of the sphere (Fig. 3.3(a)).

The 3D contour is sampled by a set of points equally distributed along a circle with a radius equal to the ball (Fig. 3.3(b)). By rotating and shifting an initial set of 3D contour points, according to the ball position, and assuming the EPM (3.1), we obtained the 2D contour in the image frame.



(a) **(b)**

Figure 3.3: Ball contour projections. (a) 3D ball points projected to obtain the 2D contour at different positions. The black line connects the projection center to the ground plane. (b) Points modeling the hypothetical ball contour (red) and points used to select inner and outer boundary pixel sets (black).

3.2.2 Observation and Motion

Given a 3-dimensional position, the previous described projection model tell us how the ball contour is going to look in the image. However, to track it, one needs to estimate the ball's location with respect to the robot. For that we use a particle filter to represent the ball's state space regarding position and velocity $\mathbf{x}_t = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$. We start by assuming a simple Markov process for the underlying dynamics of the ball specified by a transition probability, from herein denoted as motion-model, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$, and that for every time step $t > 1$ a new observation z_t about the state $\mathbf{x}_t$ is made. Given the observation history at time t by $Z_t = [z_1, ..., z_t]$ our goal is to estimate the posterior distribution $p(\mathbf{x}_t|Z_t)$ for each time step. As introduced earlier in 3.1 this can be done recursively over *Prediction* and *Update*:

$$p(\mathbf{x_t}|\mathbf{Z_{t-1}}, \mathbf{u_{t-1}}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{Z_{t-1}})p(\mathbf{x}_{t-1}|\mathbf{u_{t-1}})d\mathbf{x}_{t-1} \tag{3.2}$$

$$p(\mathbf{x_t}|\mathbf{Z_t}) \propto p(z_t|\mathbf{x}_t)p(\mathbf{x_t}|\mathbf{Z_{t-1}}, \mathbf{u_{t-1}}) \tag{3.3}$$

where $p(\mathbf{x}_{t-1}|Z_{t-1})$ is the previous estimate and $p(z_t|\mathbf{x}_t)$ is the observation model. At a given moment in time t, the particle filter represents the probability distribution of the state as a set of M weighted samples $\{\mathbf{x}_t^{(i)}, w_t^{(i)}\}_{i=1}^{M}$, such that the posterior is approximated by an empirical estimate:

$$p(\mathbf{x}_t|Z_t) \approx \sum_{i=1}^{M} w_t^{(i)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)}) \tag{3.4}$$

where $\delta(.)$ is the Dirac delta function. The estimation of the best state is computed through a discrete Monte Carlo approximation of the expectation:

$$\hat{x} \doteq \frac{1}{M} \sum_{i=1}^{M} w_t^{(i)} \mathbf{x}_t^{(i)} \tag{3.5}$$

As described in [26], the basic operation of a particle filter is to recursively estimate the posterior via a sequence of sampling, importance weighting and resampling, so the tracking algorithm performs in three steps: *Prediction, Update and Resampling*.

Prediction computes an approximation of $p(\mathbf{x}_t|Z_{t-1})$ by moving each particle according to the ball motion model. We assume a constant velocity model where the motion equations correspond to a uniform acceleration during one time step:

$$\mathbf{x}_t = \begin{bmatrix} I & (\Delta t)I \\ 0 & I \end{bmatrix} \mathbf{x}_{t-1} + \begin{bmatrix} (\frac{\Delta t^2}{2})I \\ (\Delta t)I \end{bmatrix} a_t \tag{3.6}$$

where I is the 3×3 identity matrix, $\Delta t = 1$, and a_t is a 3×1 white zero mean random vector corresponding to an acceleration disturbance. The state of the ball is expressed in a real-world robot centered coordinate system. However, if the robot is moving one should first consider its kinematic configuration. For robots moving on an horizontal plane, as considered here, roll and pitch angles can be discarded. The robot state or *pose*, $\mathbf{r_t}$, can be described by its two-dimensional Cartesian coordinates and the angular orientation $[r_x, r_y, r_\theta]^T$. The posterior distribution over the robot states is given by $p(\mathbf{r_t}|\mathbf{u_t}, \mathbf{r_{t-1}})$, where $\mathbf{r_{t-1}}$ is the robot *pose* at the previous time step and $\mathbf{u_t}$ is a motion control command given by odometry. Although technically odometry consists of sensor measurements, it is common practice to use it as control data since it's only available after the robot moved and its main information regards the actual change of the robot *pose*. One must also consider the inherent noise in robot actuation, as drift and slippage tend to induce unmeasured perturbation.

In order to consider this realistic ball motion model we need to have an inertial reference frame. Our previous work [27] describes the state of the ball expressed in the world reference frame, while the robot localization is taken into account in the observation model to compute the ball contour projection onto the image plane. However, this highly depends on the accuracy of the self-localization method, which can be troublesome. One can also just consider the robot reference frame by clearly separating the ball and the robot motion, and assume the robot does not undergo acceleration while we apply the ball motion dynamics. In fact, this assumption is valid since we only apply the ball motion dynamics when an image is captured, and we can consider that the robot is stopped while capturing it. That is, the simplification of the above described ball's motion dynamics remains, if one accounts for the robot's reference frame movement in the particles state representation $\mathbf{x_t}$. We compute the particles state given by the reference coordinate transformation yield by the robot's motion, here modeled as a Gaussian with mean $\bar{\mathbf{u}} = [\delta_x, \delta_y, \delta_\theta]^T$ and covariance matrix Σ, as

$$\mathbf{x_t} = T_{rot}\mathbf{x_{t-1}} + T_{shift}\bar{\mathbf{u}}_{\mathbf{t-1}} \tag{3.7}$$

where

$$T_{rot} = \begin{bmatrix} R_p & \mathbf{0} \\ \mathbf{0} & R_v \end{bmatrix},\ T_{shift} = \begin{bmatrix} S & \mathbf{0} \end{bmatrix}^T. \tag{3.8}$$

and R_p and R_v are the object location (x,y,z) coordinates transformation matrix and the velocity vector ($\dot{x}$,$\dot{y}$,$\dot{z}$) inverse rotation matrix respectively, while S represents the reference frame shift

$$R_p = R_v = \begin{bmatrix} \cos(\delta_\theta) & \sin(\delta_\theta) & 0 \\ -\sin(\delta_\theta) & \cos(\delta_\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}, S = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.9}$$

As such, if the tracked object's state is centered on a moving reference frame, the prediction step should compute an approximation of $p(\mathbf{x_t}|\mathbf{u_{t-1}})$ every time a new odometry measurement is obtained.

In the *Update* step, the particle's weights are updated according to the computed likelihood $p(z_t|\mathbf{x}_t^{(i)})$ for each hypothesis, from the observation model. We follow Taiana's [2] approach to compute the likelihood as a function of similarities between color histograms. We compute two YUV histograms for the inner and outer boundaries of the ball 2D projection contour and apply the Bhattacharyya [28] similarity metric. In order to track arbitrary color balls, we do not define a reference color model for the inner boundary and rely strictly on its mismatch to the outer boundary, that is the object to background dissimilarity.

The particles that have a higher weight are replicated in the *Resampling* step, and the rest of the particle set is discarded. To prevent the loss of diversity in the particle population, we use a low variance resampling technique described in [26].

In order to track the ball, one first needs to detect it. So the particle filter needs an initial distribution of particles. However, when aiming at a solution that does not rely at all on object's color, a color segmentation detection module as proposed in [2] is not appropriate. Therefore, we initialize our tracker by uniformly spreading a fixed number of ball hypothesis on the ground, in a 5 meter area surrounding the robot. This enable us to reduce the search state space, as we assume the ball is on the floor, and constrain the detection according to the camera resolution. As such, one can say we make the detection in 2D and from there we track in 3D.

4

Cooperative Perception in Mobile Sensor Networks

Contents

4.1 Information Representation

In sensor networks, the sensor measurement models are often nonlinear resulting in non-Gaussian posterior distributions. However, a parametric (e.g. Gaussian) approximation of sensors information is usually a better choice given the low computational power and low communications bandwidth it requires when sharing information. This is achieved at the cost of a limited representation of the sensors belief. Non parametric discrete approximations, such as the ones described in 3, are able to capture arbitrarily complex uncertainty, but are intractable when it comes to communicating the state distribution due to the necessity of transmitting a large sample-based representation.

The conversion of the sample-based representation to a continuous distribution requires the use of methods such as kernel density estimation, but in order to achieve efficient communication a parametrization of the probability density function is, in fact, mandatory. A mixture model provides this type of representation and can also be viewed as a type of kernel method [29]. If the kernel function of the mixture model is Gaussian, the distribution is expressed as a Gaussian Mixture Model (GMM) of the form:

$$P(\mathbf{x}) = \sum_{k=1}^{N} w_k G(\mathbf{x}|\mu_k, \Sigma_k) \tag{4.1}$$

where **x** are the observations of the random variable **X**, w_k are positive weights such that $\sum_{k=1}^{N} w_k = 1$, G is a Gaussian probability density (Gaussian mixture component) with mean μ_k and covariance Σ_k, and N is the total number of mixture components. This pdf approximation method treats the set of estimates as a sum of Gaussian probability distributions. Since the particles state is expressed in a d-dimensional space, the Gaussian distribution of state **x** with mean μ and covariance Σ is defined as:

$$P(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp^{-\frac{1}{2}[\mathbf{x}-\mu]^T \Sigma^{-1}[\mathbf{x}-\mu]} \tag{4.2}$$

For the GMM to be of practical importance both for data fusion and communications, the density estimation technique, which will lead to the parametrization of the mixture model, must be computationally fast and accurate.

Suppose one is able to extract from the particle set which particle belongs to each component $y_i = k$. This would make our GMM parametrization a lot easier, that is, we would simply compute the mean μ_k and covariance matrix Σ_k for each subset k of particles and a calculate an importance weight w_k from the number os samples assigned to each component. However, it's not possible to extract such information from the observed data and therefore this is considered the missing data for our parametrization problem. The Expectation Maximization (EM) algorithm is an

efficient iterative method to the general approach of the maximum likelihood parameter estimation in the presence of missing or unobserved data. Our main intuition while using EM is to alternate between estimating which sample from our sample-based representation belongs to which mixture component (missing data) and estimating the unknown parameters $\Theta_k = (w_k, \theta_k)$, where $\theta_k = (\mu_k, \Sigma_k)$, for each of those components. Each iteration of the EM consists of an expectation step (E-step) and a maximization step (M-step). In the E-step we compute the expected likelihood for the complete data Γ (also known as Q-function) as the conditional distribution of the missing data Y, given the current settings of parameters Θ and the observed incomplete data **X**. So, using Bayes's rule, for each mixture component k:

$$p(y_i = k|x_i, \theta_k) = \frac{p(y_i = k, x_i|\theta_k)}{p(x_i|\theta_k)} = \frac{p(x_i|y_i = k, \theta_k)p(y_i = k|\theta_k)}{\sum_{k=1}^{N} p(x_i|y_i = k, \theta_k)p(y_i = k|\theta_k)} \tag{4.3}$$

where N is the total number of mixture components and $p(x_i|y_i = k, \theta_k)$ is, in our case, the multivariate Gaussian probability density function from Eq. (4.2). One should also note that the probability of a given observation being part of a k component is actually its relative weight w_k in the mixture model. Therefore we can simplify the Q-function from Eq. (4.3) as:

$$p(y_i = k|x_i, \theta_k) = \frac{p(x_i|y_i = k, \theta_k)w_k}{\sum_{k=1}^{N} p(x_i|y_i = k, \theta_k)w_k}. \tag{4.4}$$

In the M-step we re-estimate the mixtures parameters Θ by maximizing the Q-function, see [29], [30] for the in-depth derivation. From here we can compute Θ' for each component k:

$$\mu'_k = \frac{\sum_{i=1}^{M} x_i p(y_i = k|x_i, \theta_k)}{\sum_{i=1}^{M} p(x_i|y_i = k, \theta_k)} \tag{4.5}$$

$$\Sigma'_k = \frac{\sum_{i=1}^{M} p(x_i|y_i = k, \theta_k)(x_i - \mu'_k)(x_i - \mu'_k)^T}{\sum_{i=1}^{M} p(x_i|y_i = k, \theta_k)} \tag{4.6}$$

where M is the number of total observations. The relative weight of each Gaussian mixture is given by:

$$w'_k = \frac{1}{M} \sum_{i=1}^{M} p(y_i = k|x_i, \theta_k). \tag{4.7}$$

While EM runs iteratively through these steps, improving the parameters estimation, we test the convergence of the algorithm from the observed data Log-likelihood function:

$$L(\Theta) = \ln p(\mathbf{X}|\Theta) = \ln \sum_{k=1}^{N} \sum_{i=1}^{M} p(x_i|y_i = k, \theta_k)p(y_i = k|\theta_k) \tag{4.8}$$

by maximizing the difference between the current estimate Θ and the estimation update we wish to compute Θ' (since we want $L(\Theta') > L(\Theta)$):

$$L(\Theta') - L(\Theta) = \ln(\sum_{k=1}^{N}\sum_{i=1}^{M} p(x_i|y_i = k, \theta'_k)w'_k) - \ln p(\mathbf{X}|\Theta) \tag{4.9}$$

We must assume convergence if $L(\Theta') - L(\Theta) < \psi$ for a given threshold ψ. By Jensen's inequality [29]:

$$\ln(\sum_{k=1}^{N}\sum_{i=1}^{M} p(x_i|y_i = k, \theta'_k)w'_k) \geq \sum_{k=1}^{N} w'_k \ln p(x_i|y_i = k, \theta_k). \tag{4.10}$$

Since $p(x_i|y_i = k, \theta_k)$ is given by Eq. (4.2), for all M observations:

$$\ln p(x_i|y_i = k, \theta_k) = -\frac{M}{2}\log|2\pi\Sigma_k| - \frac{1}{2}\sum_{i=1}^{M}(\mathbf{x}_i - \mu_k)^T\Sigma_k^{-1}(\mathbf{x}_i - \mu_k) \tag{4.11}$$

For computational speed, we can further simplify Eq. (4.11) according to [31] as:

$$l(\mathbf{X}|\mu_k, \Sigma_k) = -\frac{M}{2}\log|2\pi\Sigma_k| - \frac{M}{2}tr\Sigma_k^{-1}\mathbf{S} - \frac{M}{2}(\bar{\mathbf{x}} - \mu_k)^T\Sigma_k^{-1}(\bar{\mathbf{x}} - \mu_k) \tag{4.12}$$

where **S** is the covariance matrix of the observed data **X**. Therefore, we can compute:

$$L(\Theta') \geq \sum_{k=1}^{N} w_k l(\mathbf{X}|\mu_k, \Sigma_k) \tag{4.13}$$

The convergence speed however is very dependent of the initialization of Θ, and can be very slow if the initial parameters are particulary bad compared to the true values. Clustering algorithms based on the k-means implementation have proven to have reasonable results in the parameter initialization, ensuring only a few iterations before convergence [22]. Table 4.1 depicts initialization (lines 1 to 7) and the rest of the previously described EM implementation in pseudo-code.

4.2 Mobile Cooperative Sensor Model

More than deriving and applying an efficient multi-sensor data fusion technique, such as the ones described earlier in Section 2.3, our work aims at developing an adequate model to fuse and improve mobile sensors belief in a decentralized network operating on dynamic environments. Unlike in static environments, the state of the targets can change over time so we can only combine sensor observations gathered simultaneously. Furthermore, in fully DDF systems the topology of

Table 4.1: Expectation Maximization algorithm for GMM parameter estimation

	Algorithm **ExpectationMaximization**($\mathbf{X}, nModes$)
1:	$cluster =$ **k-means**($\mathbf{X}, nModes$)
2:	
3:	for $k = 1$ to $nModes$ do
4:	$\mu_k =$ **mean**($cluster_k$)
5:	$\Sigma_k =$ **cov**($cluster_k$)
6:	$w_k =$ **size**($cluster_k$)/**size**($\mathbf{X}$)
7:	endfor
8:	
9:	while $(l_n - l_{n-1}) > \psi$
10:	for $k = 1$ to $nModes$ do
11:	$E - step$: compute $p(y_i = k \mid x_i, \theta_k)$
12:	endfor
13:	
10:	for $k = 1$ to $nModes$ do
13:	$M - step : \mu'_k = \frac{\sum_{i=1}^{M} x_i p(y_i=k \mid x_i, \theta_k)}{\sum_{i=1}^{M} p(x_i \mid y_i=k, \theta_k)}$
14:	$\Sigma'_k = \frac{\sum_{i=1}^{M} p(x_i \mid y_i=k, \theta_k)(x_i-\mu'_k)(x_i-\mu'_k)^T}{\sum_{i=1}^{M} p(x_i \mid y_i=k, \theta_k)}$
15:	$w'_k = \frac{1}{M} \sum_{i=1}^{M} p(x_i \mid y_i = k, \theta_k)$
16:	endfor
17:	$l_n = \sum_{k=1}^{nModes} w_k l(\mathbf{X} \mid \mu'_k, \Sigma'_k)$
18:	endwhile
19:	return μ'_k, Σ'_k, w'_k

the network is unknown and so we are forced to deal with the problem of ”double counting”. This implies the removal of common information between the received and the local estimate in order to avoid an over-confident final estimate.

So the decentralized sensor fusion typical approach is to build one single estimate of the target, regardless of whether it’s being tracked by the local sensor or not, and always assume that in the worst case we are improving the local error resulting in a more accurate estimate. This is perfectly valid if the sensor is not moving or if we can assume a highly accurate localization system, which is not often the case. Even so, previous work shows that the robot localization uncertainty can be taken into account while propagating it to the sensors belief [20], achieving an accurate and more consensual team estimate. Still, we are implicitly assuming that the result of the fusion process is more relevant than the local sensor observations. We propose a different approach that consist of not taking other sensors beliefs for granted, and instead use them as if they were observations gathered by the local sensor (virtual observations).

From the previously described particle filter based world perception framework in chapter 3, we present herein a cooperative perception model that copes both with a local sensor-distributed estimate of the object and a fused team estimate, naturally deals with the correlation between common information and can be used to improve self-localization. The model, based in sequential Bayesian filtering representation, is illustrated in fig. 4.1.

In the Local Filter, observations are made and used to compute the likelihood over the sensor

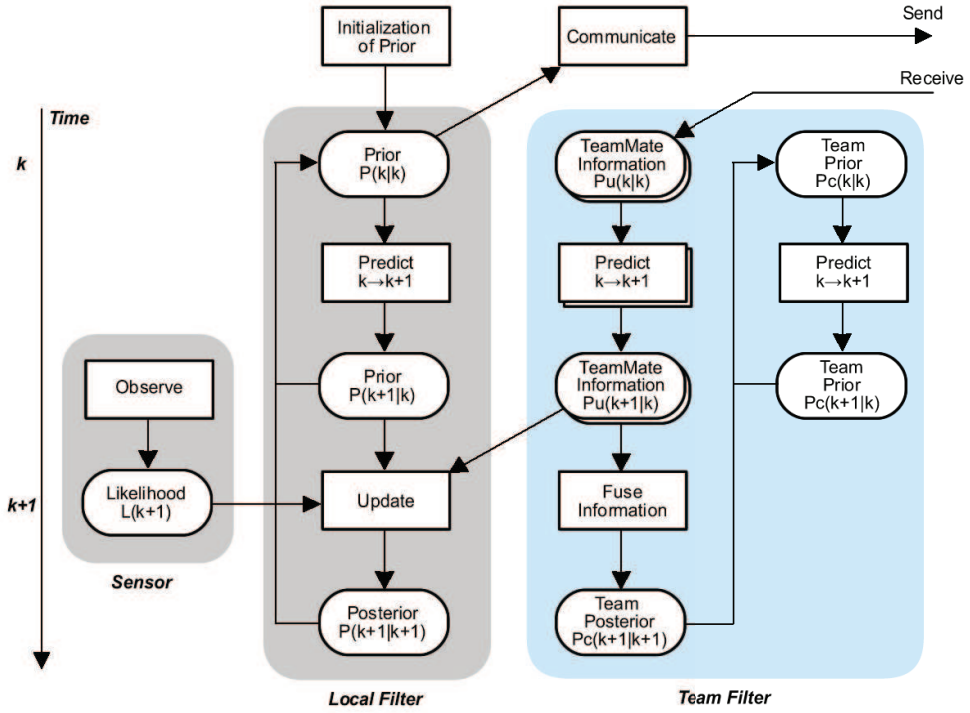


Figure 4.1: Mobile Cooperative Sensor Model

model, which then is multiplied by the prior belief in the update step. Both the local prior, predicted from the local posterior over the previous state, and the team prior, predicted from the received posterior distributions of the teammates, are concurrently computed at each robot. This way, the other robots information will only influence the prior belief and posterior will be given according to the local sensor measurement model. The Team Filter will fuse all the teammate beliefs into one parallel target estimate and, while it doesn't receive new information, will keep predicting over the previous state from the last received ball velocity. This parallel team estimate is to be used only in critical conditions when the target is out of the sensor field of view. Keeping these two (local and team) estimates of the same target allows to improve robot localization when its uncertainty is too high, since we can infer a prior belief with respect to the target location on the world frame and the robot frame.

4.2.1 Local Filter

In our case we are interested in detecting and tracking a ball with the particle filter already described in Section 3.2 so we must implement the Local Filter as a discrete Bayes filter. In the update step, we sample from the Bayes prior distribution, denoted from here and henceforth as the proposal distribution $\overline{bel}(\mathbf{x}_t)$, and our goal is that the weighted particle set approximates the Bayes posterior, denoted from here and henceforth as the target density $bel(\mathbf{x}_t)$. Upon resampling, the particles are distributed according to the posterior:

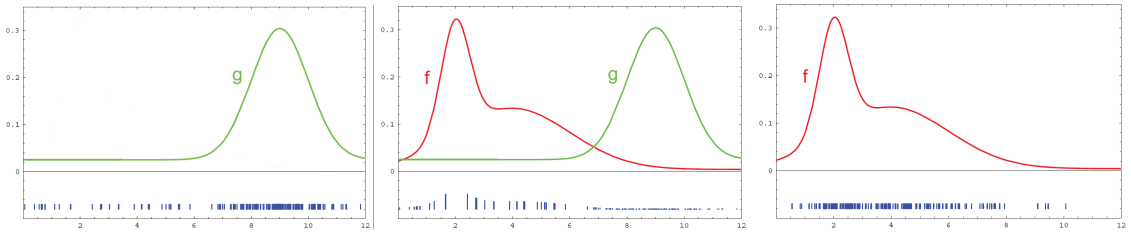


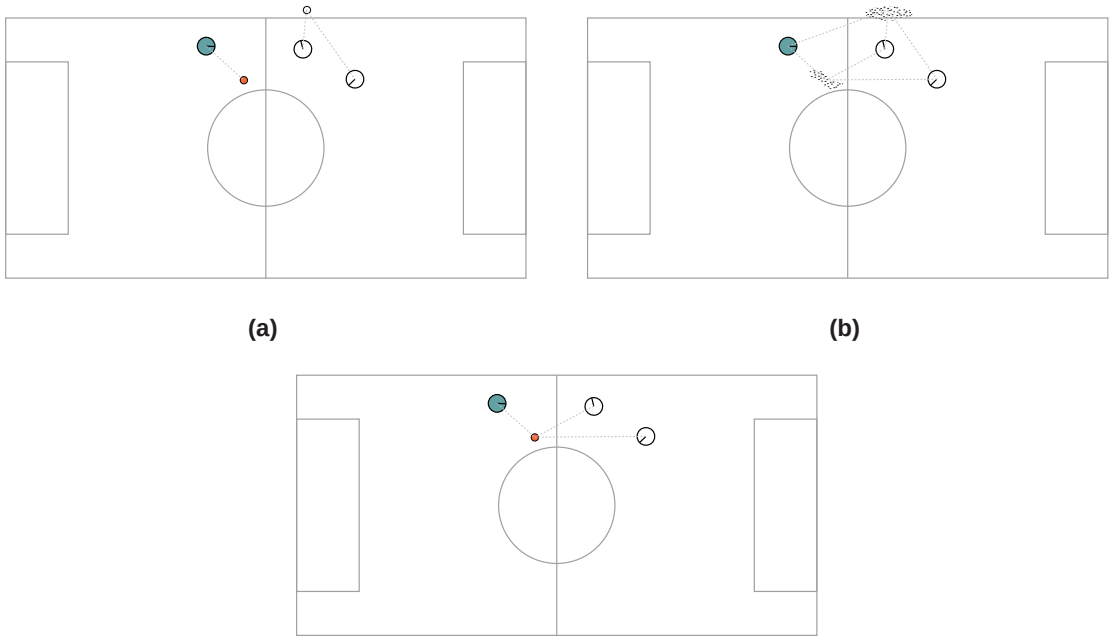
Figure 4.2: Resampling step. (a) We sample from a proposal distribution *g* that comprises our and the other robots ball belief, where there is likely to exist common correlated information. (b) Sample weights are updated according to the observation model. (c) Particles are redistributed according to their weights and the new density distribution approximates the target density *f*. Therefore eliminating the risk of producing an over confident estimate since it depends always on the weight of the samples given by the local observation model, rather than the density of the proposal distribution.

$$bel(\mathbf{x}_t^{[m]}) = \eta p(z_t|\mathbf{x}_t^{[m]})\overline{bel}(\mathbf{x}_t) \tag{4.14}$$

where $p(z_t|\mathbf{x}_t^{[m]})$ is the probability of measurement z_t under the mth particle $\mathbf{x}_t^{[m]}$. The target density is then transformed in a compact GMM representation and passed on to the other robots. When it is received, new samples will be drawn from it contributing for the proposal distribution. The ability to sample is not given for arbitrary distributions, however, since our distributions can actually be decomposed in a sum of Gaussians, we can draw a random vector $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)^T$ from each bivariate component k with mean μ_k and covariance matrix Σ_k from:

$$\mathbf{x}_k^{[n]} = A_k v^{[n]} + \mu_k \tag{4.15}$$

where v are n independent samples drawn from $N(0, I_2)$ and A_k is the Cholesky decomposition of Σ_k, such that $\Sigma_k = AA^T$. For each new particle $\mathbf{x}^{[n]}$ we then calculate the importance factor w as described in the ball tracking update step, Section 3.2. As such, samples generated from received GMMs that do not follow the local observation model will have a low likelihood and will be discarded on resampling. One should point out at this point that we are eliminating the risk of generating an over confident estimate from common correlated information after the resampling step, since the transmitted target distribution $bel(\mathbf{x}_t)$ only incorporates the sender local measurements z_t. The resampling step is illustrated in Fig. 4.2 taken from [26] for a monodimensional example.



(a) (b)

(c)

Figure 4.3: Local Filter running the ball tracking distributed particle filter. (a) Blue robot is tracking the ball while its teammates are tracking a fake similarly shaped ball. (b) After communicating their target density, all robots have a similar proposal distribution. (c) On Update, the importance factor is re-calculated for each of the particles drawn from the proposal distribution and only the best ones ”survive”.

4.2.2 Team Filter

In the Team Filter we receive GMM representations, from all teammates, of the ball's posterior in the world frame. As susch, the robot's odometry control measurement take no part in the prediction step, only the ball velocity is relevant. Here we assume the ball velocity $v_t = [\dot{x}, \dot{y}]$ is constant between each time step, and predict based on a received ball velocity estimate for each GMM:

$$\mu_t = \mu_{t-1} + \frac{v_t}{\Delta t} \tag{4.16}$$

After fusing all the received GMM into a posterior team estimate, if no new information arrives, prediction will produce a less and lesser accurate estimation since no update step occurs.

Regarding information fusion, the Covariance Intersection (CI) filter yields consistent estimates to the problem of combining different Gaussian random vectors with unknown correlation between them [32]. This can be extended to a GMM Covariance Intersection algorithm as in [22], by performing CI between each of the mixture components. The fusion between the ith component of a GMM and the jth component of another GMM will result in a Gaussian mixture with $N \times N$ components, such that:

$$\Sigma_{ij}^{-1} = \gamma\Sigma_i^{-1} + (1-\gamma)\Sigma_j^{-1} \tag{4.17}$$

$$\mu_{ij} = \Sigma_{ij}(\gamma\Sigma_i^{-1}\mu_i + (1-\gamma)\Sigma_j^{-1}\mu_j) \tag{4.18}$$

$$w_{ij} = \frac{1}{N}(\gamma w_i + (1-\gamma)w_j) \tag{4.19}$$

where $0 \leq \gamma \leq 1$ is a weighting parameter to minimize the determinant of the result.

When associating data in distributed systems, an incorrect association decision leads to an incorrect fusion estimate, therefore one needs to have the ability to measure agreement among disparate sensors before fusing their observations. We assume there's an agreement between two received GMMs G_1 and G_2 if:

$$D(G_1, G_2) \leq \xi \tag{4.20}$$

where D(.) is a distance metric between both GMMs and ξ is a positive threshold. A distance measure between Gaussian distributions can be defined as Kullback-Leiber distance [33], Bhattacharyya distance [28] and others. However there's no analytical solution of computing these measures to evaluate the distance between Gaussian mixture models. Therefore, we take Beigi et al. [34] approach to measure distances between collections of distributions in speech recognition, and define our measure of divergence between GMMs as:

$$D(G_1, G_2) = \frac{\sum_{i=1}^{N} W_i^1 + \sum_{j=1}^{N} W_j^2}{\sum_{i=1}^{N} c_i + \sum_{j=1}^{N} c_j} \tag{4.21}$$

Consider the matrix of distances between $N \times N$ mixture componentes:

$$T = \begin{bmatrix} d_{11} & d_{12} & ... & d_{1N} \\ d_{21} & d_{22} & ... & d_{2N} \\ ... & ... & ... & ... \\ d_{N1} & d_{N2} & ... & d_{NN} \end{bmatrix} \tag{4.22}$$

W_i^1 is the minima of the elements in the row times the row number c_i. Likewise, W_j^2 is the minima of the elements in the column times the column number c_j. We can compute d_{ij} from the above metrics for Gaussian distributions. We choose to apply the Bhattacharyya distance, given by:

$$d_{ij} = \frac{1}{8}(\mu_j - \mu_i)^T\Lambda^{-1}(\mu_j - \mu_i) + \frac{1}{2}\ln\frac{|\Lambda|^{-1}}{\sqrt{|\Sigma_i||\Sigma_j|}} \tag{4.23}$$

for multivariate Gaussian distribution, where $\Lambda = \frac{\Sigma_i + \Sigma_j}{2}$, to compute the distance between components of different mixtures.

So when there is agreement, after fusing the received GMMs with CI we can compute the ball localization from a weighted mean average of all the components means that describe the final Gaussian mixture team estimate:

$$BallTeamEstimate = \sum_{k=1}^{N \times N} w_k \begin{bmatrix} \mu_x & \mu_y \end{bmatrix} \tag{4.24}$$

4.2.3 Improving Self-Localization

In respect to the global localization problem, most state-of-the art algorithms cannot guarantee a full proof method that never fails [26]. The detection of a failure and the ability to recover from it is essential for truly autonomous robots. When the robot believes it knows where it is, while in fact it does not, it can usually figure out that something's wrong by the disparity between the sensor measurements and the sensor model. However, as presented in chapter 3.1 this can also lead to false kidnapping situations when dealing with unprecise limited-view sensors that have to cope with unsteady robot movements. To prevent this, one can also test the disparity between a common observed target, that is the disagreement between the ball estimate observed by the robot and the ball estimate of the teammates. So if the MCL estimate has a low likelihood and there's disagreement between the Local Filter and the Team Filter estimate, the robot is most likely to be lost.

The normal approach to recover from such situation in MCL consists in gradually augmenting the proposal distribution by systematically adding more and more particles until better observation likelihoods can be obtained. Two major drawbacks can compromise this approach. One is the large amount of computational power required to draw and test samples from an augmented proposal distribution that can comprise the entire state space, which basically means a restart in the global localization problem. The other drawback is the inability to deal with local maxima that are present in symmetric environments, such as the RoboCup field.

Instead, one can now see the problem as feature-based map localization with known correspondence, that is $p(\mathbf{x}_t|f_t^i, c_t^i, m)$, where f_t denotes a given feature that has a correspondence c_t in a list of landmarks m. Let's consider the ball as a landmark m_1. If some other robots are localized and tracking the ball, the coordinates $m_{1,x}$ and $m_{1,y}$ of our landmark in the world frame of the map are given by the Team Filter estimate. If the lost robot is tracking the ball relative to its local coordinate frame (Local Filter), it can make new guesses of its own whereabouts for it now knows it may be on a circle around the landmark. These new guesses represent new poses that incorporate the sensor measurement $p(f_t^i|c_t^i, \mathbf{x}_t, m)$. Basically, we can assume the robot is completely lost and therefore the prior $p(\mathbf{x}_t|c_t^i, m)$ is uniform. This assumption leads to:

$$\begin{aligned} p(\mathbf{x}_t|f_t^i, c_t^i, m) &= \eta p(f_t^i|c_t^i, \mathbf{x}_t, m) p(\mathbf{x}_t|c_t^i, m) \\ &= \eta p(f_t^i|c_t^i, \mathbf{x}_t, m) \end{aligned} \tag{4.25}$$

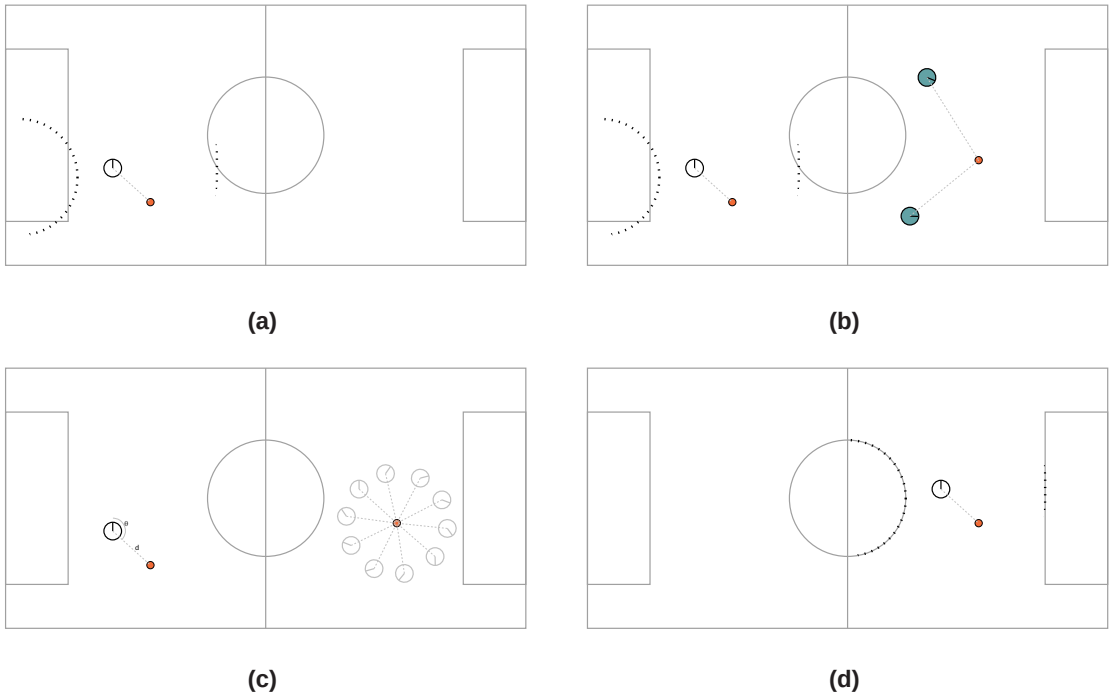


Figure 4.4: Improving robot localization. (a) The robot tracks the ball in the local frame (Local Filter) but its localization sensor observations (line detection) have a large mismatch to the sensor model, indicating a low likelihood posture estimate. (b) Teammates communicate their target density in the world frame and the robot is able to compute the ball team estimate (Team Filter) realizing it disagrees with the local one. (c) From its known pose with respect to the ball local estimate, it computes new pose hypothesis (proposal distribution) with respect to the ball team estimate. (d) A better pose that maximizes the sensor model likelihood is achieved and the Local and Team Filter have reached an agreement.

from where we concluded that sampling from $p(\mathbf{x}_t|f_t^i, c_t^i, m)$ can, in this particular case, be achieved from $p(f_t^i|c_t^i, \mathbf{x}_t, m)$. As so, we only add a limited amount of new sample poses that derive from a common target observation to the MCL proposal distribution. The complete process is illustrated in fig. 4.4.

5

Results

Contents

5.1 Experimental setup

This work was implemented in the ISocRob software architecture [1] in order to demonstrate a real time application of the methods proposed with robots playing in the RoboCup Middle Size League. The architecture is based on the Active Object design pattern. Each of these objects retains their own execution context and execution flow, allowing to add or remove objects without major impact over the rest of the system. The current low-level (ATLAS, see fig. 5.1) information flow however, is not very well suited to the multisensor fusion task at hand. The issue here is that services running on a given layer (e.g. BallFusion running on Information Fusion layer) are not allowed to communicate with services in the same layer or beneath it. Because of that, this implementation is forced to differ from the exact model methodology presented in Section 4.2 and illustrated in fig. 4.1. We work around it to make it possible for the self-localization service (MCL) to directly communicate with the Team Filter, and there we synchronize both the localization data and the tracking ball data, in order to build a ball particle set in the world frame. It is over this set of particles that we run EM to compute the *BallGMM* that is then communicated to other robots. Note that in the presented model (Fig. 4.1) this occurs in the Local Filter rather than in the Team Filter. The communication from Team Filter to Local Filter was solved recurring to a common information repository (WorldInfo). This way the Local Filter is able to retrieve directly from the WorldInfo the teammates *OtherRobotsBallGMM* in order to implement the distributed particle filter.

To achieve maximum processor performance, the implementation was done in C++ with extensive use of Intel Performance Primitives (IPP) for optimized vector and matrices operations and Intel Math Kernel Library (MKL) for statistics procedures. The Local Filter and the information framework are concluded, but some work still remains on the Team Filter at this point so we also used Matlab to do some offline processing and validate the achieved results.

5.2 Ball Tracking

5.2.1 Arbitrary Color Ball Tracking with a Moving Robot

In this experiment the omnidirectional robot tracks a moving ball, moving while attempting to catch it. We carried out the experience with an ordinary soccer ball, mainly white colored, as one can see in Fig. 5.2a, but other colored balls, e.g. orange, could and have been used [27].

Images on the robot where acquired using an omnidirectional camera with a dioptric setup at 10fps. Odometry motion control measurements where obtained at 25fps and we used only 600 particles in the tracker. The results are visible in Fig. 5.2b where one can perceive the robot path while pursuing the detected ball (arbitrary trajectory) in a global reference frame (the soccer field centered frame).

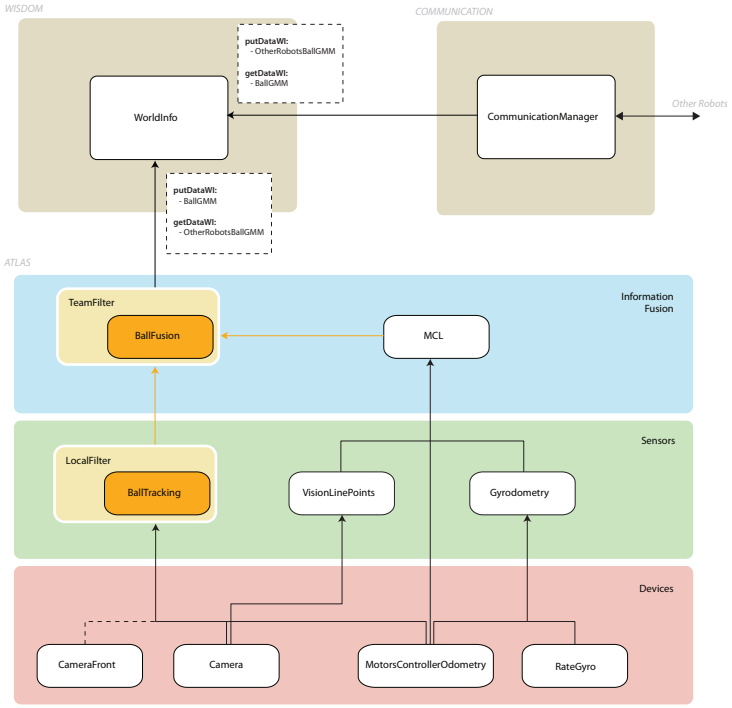


Figure 5.1: Cooperative Perception Model implementation in ISocRob software architecture

Table 5.1: Average execution time while computing GMMs with our EM implementation for 12000 particles

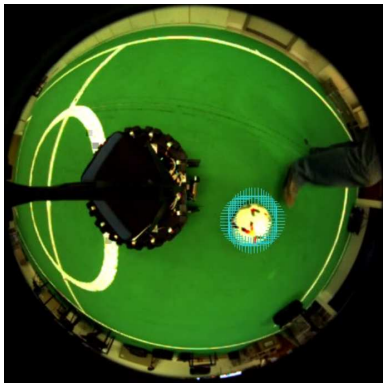
Number of mixture components	**1**	**2**	**4**	**10**
Time taken [seconds]	0.0246	0.0690	0.1131	0.1924

5.3 Cooperative Perception

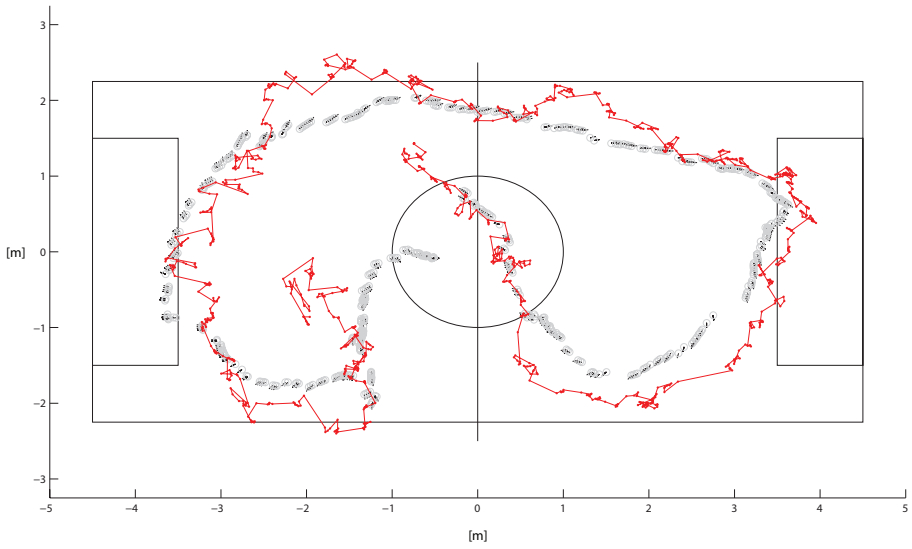
5.3.1 Generating Compact Information Representations

In this experiment we test the accuracy of the GMMs approximations with EM. First we placed the robot stopped near the ball and tested the same scenario (figure 5.3a) with a different number of mixture components (1, 2, 4 and 10) in order to achieve an good number od modes given the EM time to converge and the final approximation to the particle representation. The results are visible in figures 5.3, 5.4, 5.5 and 5.6 and the average EM run time was recorded in table 5.1. Afterwards (figure 5.7), we placed the robot in a more uncertain location (not so close to the lines) and further away from the ball to see how it's pose and sensor uncertainty are propagated to the GMM representation.

All the processing was made online with 600 particles in the ball tracker and a minimum of 20 particles in the MCL.



(a)



(b)

Figure 5.2: Tracking a white ball with a moving robot. (a) Detection of a white ball based on the object-to-background dissimilarity, where the blue crosses mark the pixels used to build the background histogram. (b) Plot of the paths of robot and ball. The robot starts in the middle circle facing opposite to the ball. The gray circles represent the robot's *pose* while the red dots represent the ball localization, here in a 2D representation only.

Table 5.2: Distance measurements between the GMMs received by robot omni1

$\mathbf{D(G_2, G_3)}$	9.6880
$\mathbf{D(G_3, G_4)}$	200.5146
$\mathbf{D(G_2, G_4)}$	162.7252

5.3.2 Fusing Data with Agreement

In this experiment two robots are able to locate the ball, while a third robot (the goalkeeper) cannot (see Fig. 5.8). The robots tracking the ball compute their GMM approximation and broadcast it to others. As the goalkeeper receives the teammates GMM estimates, it first tests it to see if there's agreement, and if ok proceeds to compute a team estimate by fusing the GMMs with Covariance Intersection.

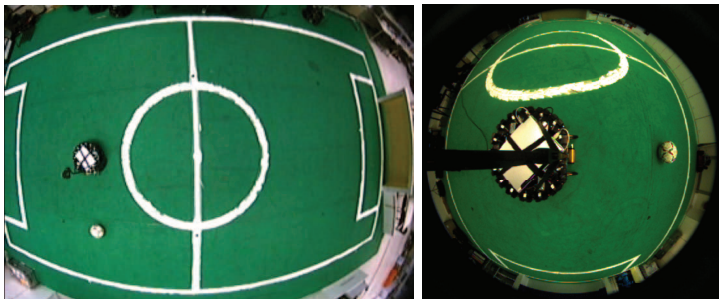
We set the EM to compute 4 components of Gaussian mixtures since it provides an acceptable approximation of the particle set in a short amount of time, therefore consuming less cpu resources. Some processing was made partially off-line in Matlab, namely the GMMs disagreement measuring and the GMM Covariance Intersection. We set the threshold $\xi = 30$ from Eq. (4.20) in order to decide if two observers disagree.

It is clear that both robots are in agreement and our GMM distance measuring proves it: $D(G_1, G_2) = 10.4114 \leq \xi$. The final fused GMM is shown in Fig. 5.8f and represents the goalkeeper ball team estimate computed in the Team Filter. We can see from Fig. 5.8g that it yields consistency to the other robots estimates in fig. 5.8c and Fig. 5.8e.

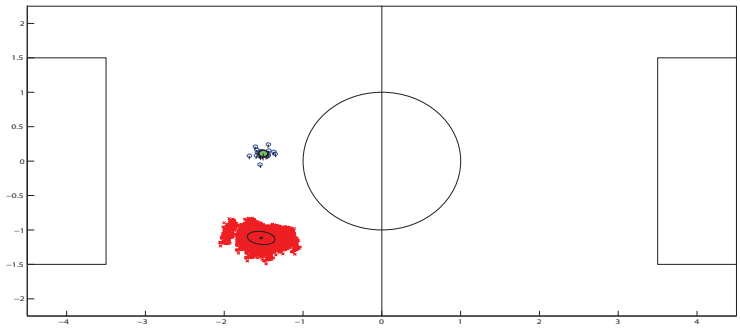
5.3.3 Fusing Data with Disagreement

In order to test data association with disagreement, another robot as placed on the field (Fig. 5.9a). Although its able to track the ball, this robot (omni4) is not able to localize itself correctly on the field. As such, it broadcast a GMM approximation of its erroneous ball localization belief, since it is corrupted by its self localization belief (Fig. 5.9b).

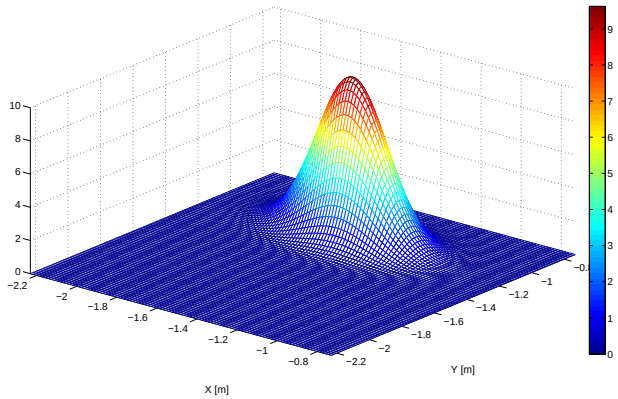
We show the results of the fusion estimate made by robot omni1 (Fig. 5.9g, 5.9h), which is not able to see the ball at all. The decision on which GMMs to fuse is based on the disagreement measurement Eq.(4.20) with $\xi = 30$. The computed distances between each of the received GMMs are shown is Table 5.2.



(a) (b)

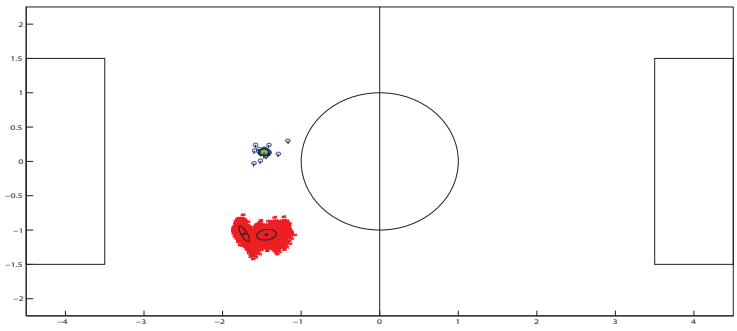


(c)

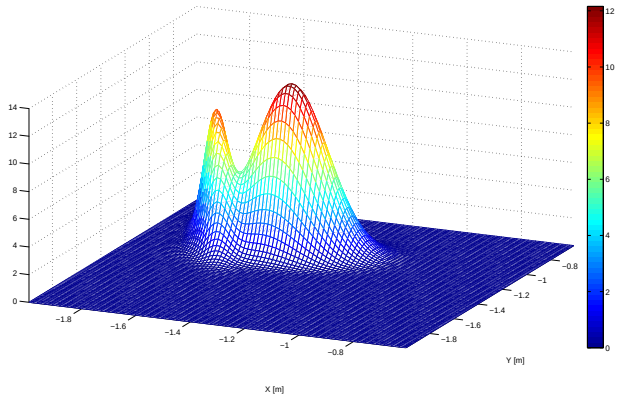


(d)

Figure 5.3: Computing GMMs: Single Gaussian. (a) Top field camera view. (b) Robot view. (c) Standard deviation ellipse of the computed gaussian, ball particle set (red crosses) and robot pose. (d) Approximation of the ball particle representation.

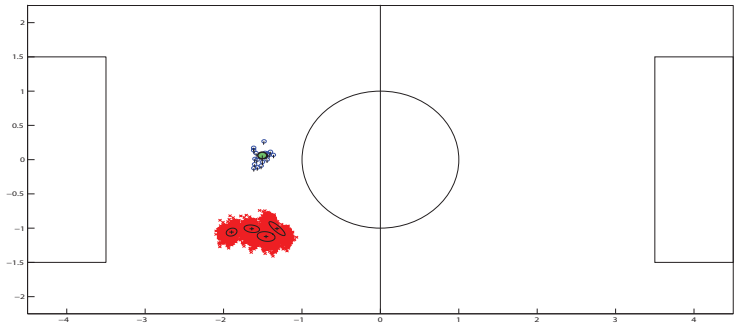


(a)

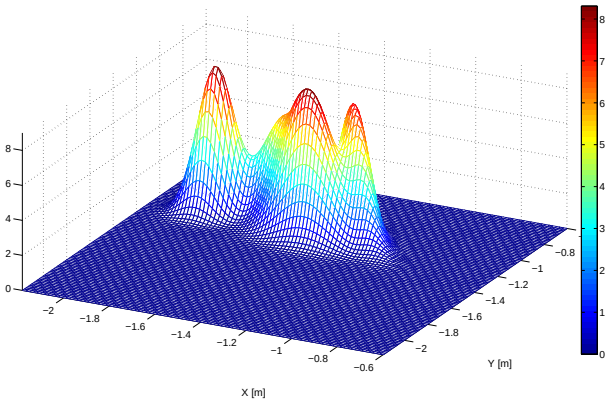


(b)

Figure 5.4: Computing GMMs: 2 mixture components. (a) Standard deviation ellipses of the computed modes. (c) Approximation of the ball particle representation.

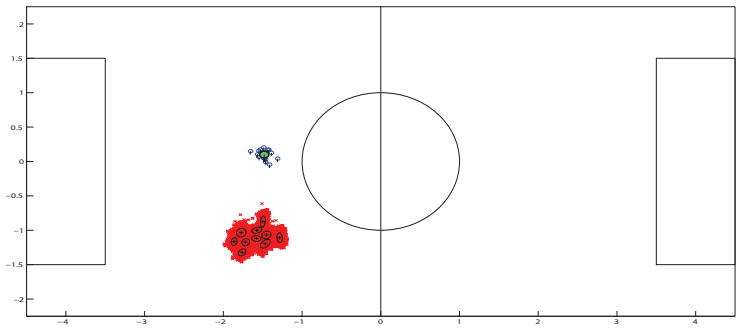


(a)

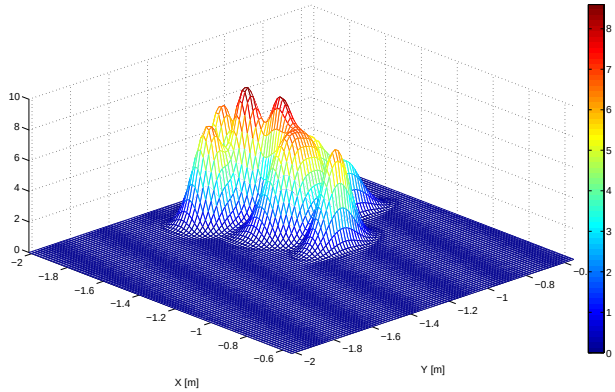


(b)

Figure 5.5: Computing GMMs: 4 mixture components. (a) Standard deviation ellipses of the computed modes. (c) Approximation of the ball particle representation.

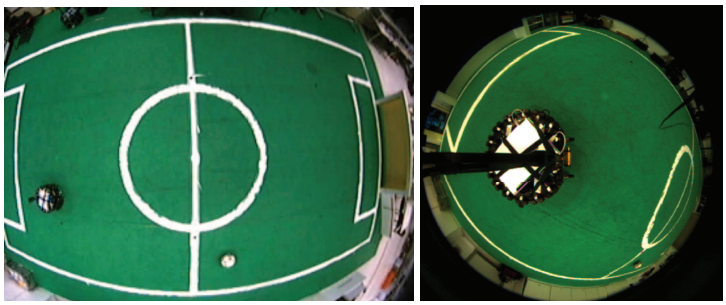


(a)

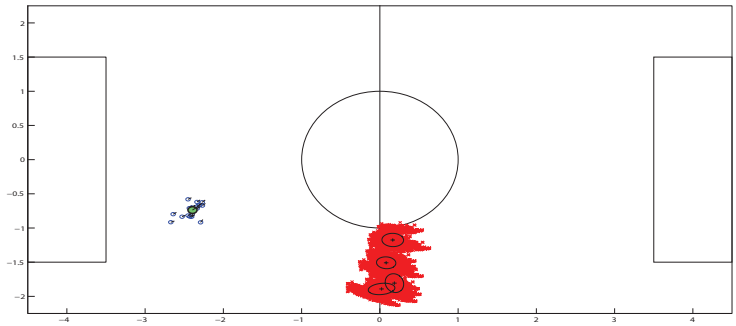


(b)

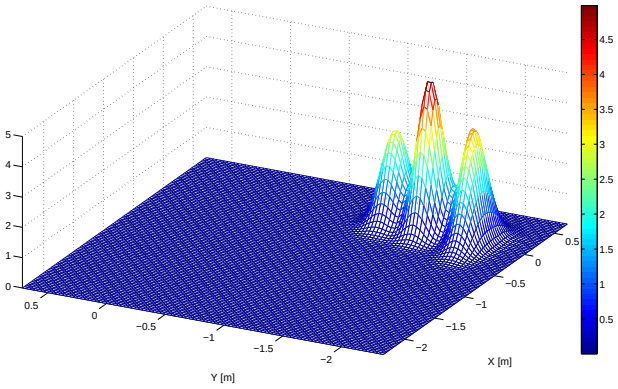
Figure 5.6: Computing GMMs: 10 mixture components. (a) Standard deviation ellipses of the computed modes. (c) Approximation of the ball particle representation.



(a) (b)

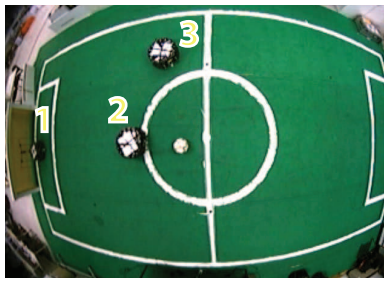


(c)

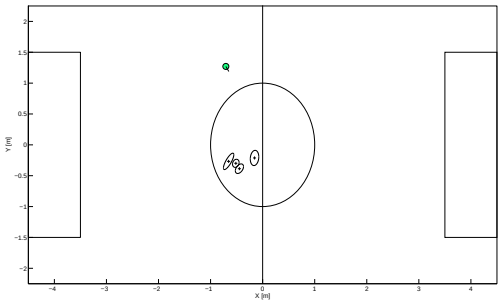


(d)

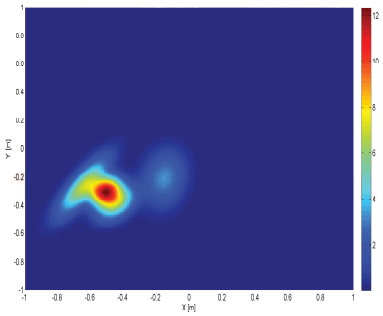
Figure 5.7: Computing GMMs: Propagating uncertainty. (a) Top field camera view. (b) Robot view. (c) Standard deviation ellipse of the computed modes. The robot pose is less certain (MCL particles are more scattered) and because it's also far from the ball, the error is propagated to the ball target density (red crosses). (d) Approximation of the ball particle representation. It is clear that the error is captured by the GMM.



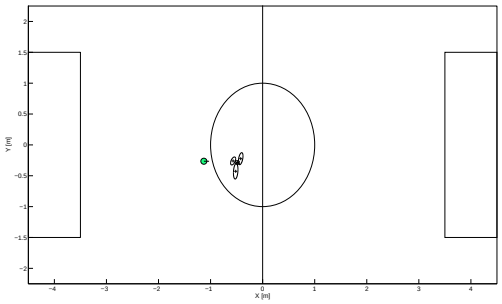
(a)



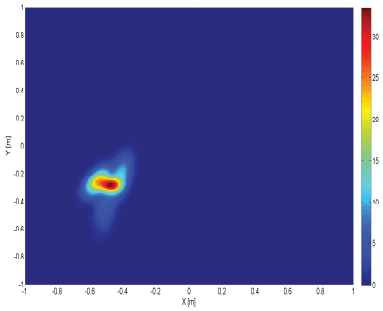
(b)



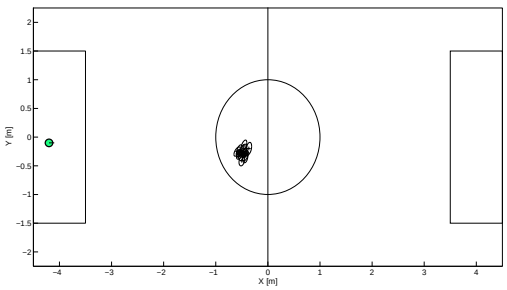
(c)



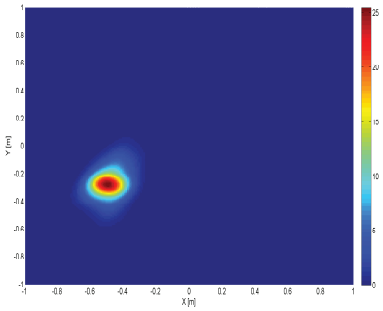
(d)



(e)



(f)



(g)

Figure 5.8: GMM Data Fusion with Agreement. (a) Top field camera view. (b-c) Robot omni2 tracks the ball, computes it's GMM and broadcast it. (d-e) Robot omni3 that tracks the ball, and also computes it's GMM and broadcast it. (f) The goalkeeper (omni1) tests received GMMs for disagreement and computes GMM CI. (g) The final fused GMM is consistent.

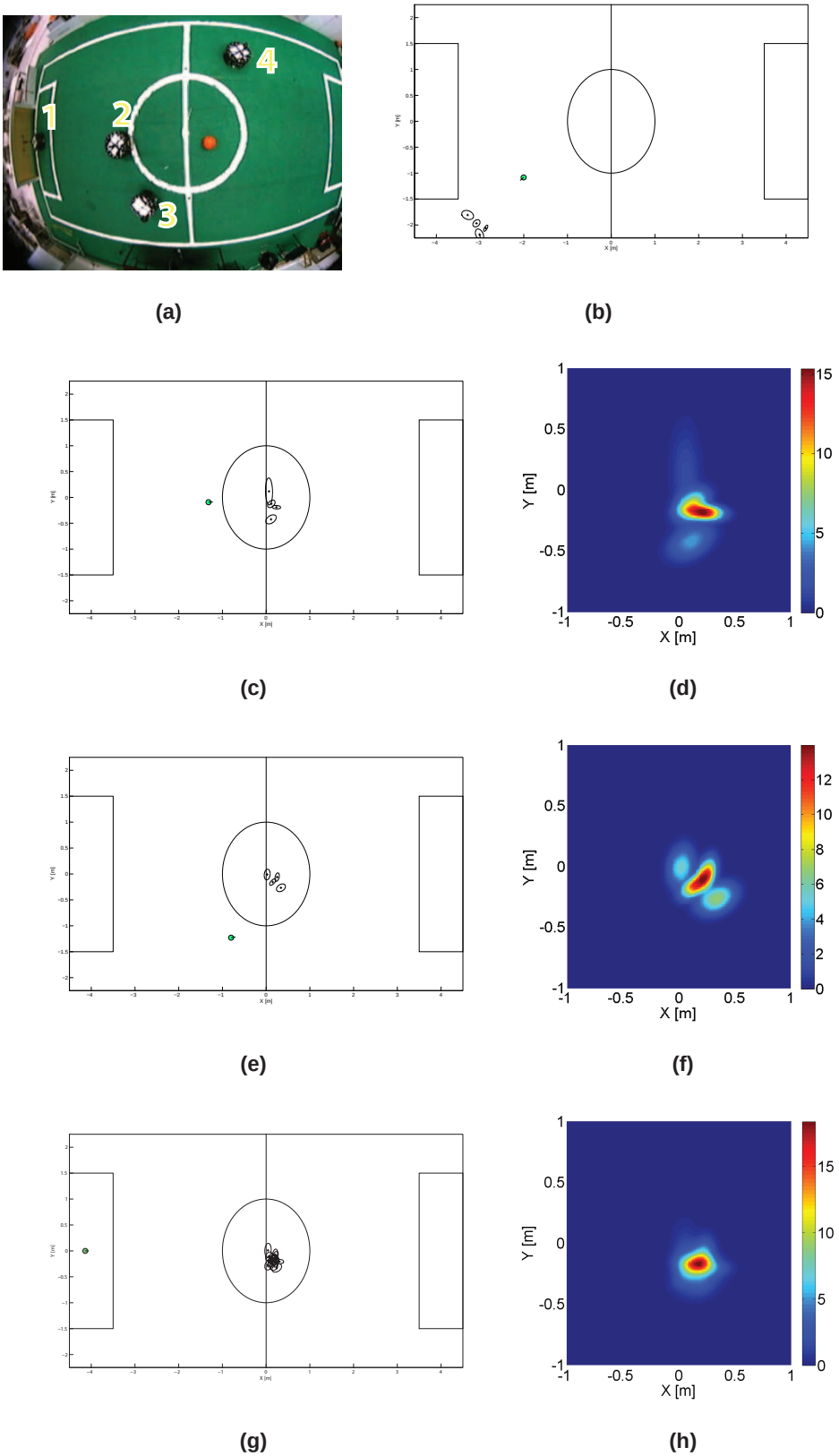


Figure 5.9: GMM Data Fusion with Disagreement. (a) Top field camera view. (b) Robot omni4 tracks the ball and broadcasts its GMM, but is not well localized. (c-f) Robots omni2 and omni3 track the ball, compute their GMMs and broadcast it. (g-h) The goalkeeper (omni1) tests received GMMs for disagreement and computes GMM CI only for those that are in agreement.

6

Conclusions and Future Work

Contents

6.1 Conclusions

We presented a cooperative sensor fusion model based on a particle filter perception framework, for mobile robots operating in dynamic environments. It was designed to cope with the RoboCup MSL environment and it was implemented in ISocRob omnidirectional robots. We aim at taking advantage of a team of sensors to detect the ball on the field at all time.

For that we implemented a 3D shaped-based ball tracker that comprises a realistic dynamic motion model. The system is based on particle filters and also comprises an observation model that allow us to compute the likelihood of a ball hypothesis, given the ball shape model, the projection model for the omnidirectional camera and an acquired image. To acquaint for the robot motion in the tracker we apply the inverse dynamics of the robot to the particle filter. Experiments show that we are able to track arbitrary color balls in 3D with a moving robot.

We also presented a framework for representing and measuring disagreement of sensor information based on Gaussian Mixture Models. This representation allows to capture arbitrary complex uncertainty from nonlinear observation models, yet it's parametrization is simple and takes no overhead in communications. We implemented the Expectation Maximization algorithm for GMM parameter estimation to approximate the sample based ball posterior distribution.

The implemented cooperative perception model takes advantage of the GMM representation in two distinct forms. One is to improve the local ball particle filter in a distributed fashion way by injecting new particles drawn directly from the received GMMs. The other is to compute a ball team estimate directly from the received GMMs target distribution with Covariance Intersection.

6.2 Future Work

As for future work, we propose to:

- concluded the cooperative perception model implementation in ISocRob software architecture;
- extend the ball tracker with a more complex motion model that models the ball being kicked over the air;
- extend the ball tracker to work with simultaneous cameras with different setups, and implement it in the ISocRob robots using the omnidirectional camera and the front camera at the same time;
- improve the ball tracker ego-motion compensation with gyrodometry to correct the drift error;
- dynamically compute which is the most suited number of Gaussian mixture components to approximate a given distribution, using KD-trees like Ihler and Willsky [35];

- use GMM Covariance Union in data fusion to prevent an excessive number of final components after Covariance Intersection

Bibliography

[1] H. Costelha, N. Ramos, J. Estilita, J. Santos, M. Tajana, J. Torres, T. Antunes, and P. Lima, “Isocrob 2007 team description paper,” Instituto Superior Técnico, Tech. Rep., 2007.

[2] M. Taiana, “3d model-based tracking with one omnidirectional camera and particle,” Master’s thesis, Laboratorio di Intelligenza Artificiale e Robotica, Politecnico di Milano, 2007.

[3] J. L. Azevedo, N. Lau, G. Corrente, A. Neves, M. B. Cunha, F. Santos, A. Pereira, L. Almeida, L. S. Lopes, P. Pedreiras, J. Vieira, D. Martins, N. Figueiredo, J. Silva, N. Filipe, and I. Pinheiro, “Cambada2008: Team description paper,” University of Aveiro, Tech. Rep., 2008.

[4] O. Zweigle, U.-P. Kappeler, T. Ruhr, K. Haussermann, R. Lafrenz, F. Schreiber, A. Tamke, H. Rajaie, A. Burla, M. Schanz, and P. Levi, “Cops stuttgart team description 2007,” University of Stuttgart, Tech. Rep., 2007.

[5] T. Hafner, S. Lange, M. Lauer, , and M. Riedmiller, “Brainstormers tribots team description,” University of Osnabruck, Tech. Rep., 2008.

[6] M. Lauer, S. Lange, and M. Riedmiller, “Modeling moving objects in a dynamically changing robot application,” in KI 2005: Advances in Artificial Intelligence, 2005, pp. 291–303.

[7] H. F. Durrant-Whyte, “Sensor models and multisensor integration,” International Journal of Robotics Research, vol. 7, no. 6, pp. 97–113, 1988.

[8] A. G. Mutambara, Decentralized estimation and control for multisensor systems. CRC Press, 1998.

[9] M. Dietl, J.-S. Gutmann, and B. Nebel, “Cs freiburg: Global view by cooperative sensing,” in International RoboCup Symposium, 2001.

[10] M. Dietl, J. S. Gutmann, and B. Nebel, “Cooperative sensing in dynamic environments,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2001.

[11] F. d’Agostino, A. Farinelli, G. Grisetti, and D. Iocchi, L. andNardi, “Monitoring and information fusion for search and rescue operations in large-scale disasters,” in International Conference on Information Fusion, 2002.

[12] F. Zhao, J. Shin, and J. Reich, “Information-driven dynamic sensor collaboration for tracking applications,” IEEE Signal Processing Magazine, vol. 19, no. 2, pp. 61–72, 2002.

[13] T. Suzuki, N. Tomoyasu, M. Takafashi, and K. Yoshida, “Eigen keio univ. team description,” Keio University, Tech. Rep., 2008.

[14] N. Lau, L. S. Lopes, and G. A. Corrente, “Cambada: Information sharing and team coordination,” in Robótica 2008, 2008.

[15] A. Ferrein, L. Hermanns, and G. Lakemeyer, “Comparing sensor fusion techniques for ball position estimation,” in RoboCup 2005 Symposium, 2005.

[16] A. W. Stroupe, M. C. Martin, and T. Balch, “Merging probabilistic observations for mobile distributed sensing,” Carnegie Mellon University, Tech. Rep., 2000.

[17] P. Pinheiro and P. Lima, “Bayesian sensor fusion for cooperative object localization and world modeling,” in 8th Conference on Intelligent Autonomous Systems, 2004.

[18] A. Cai, T. Fakuda, and F. Arai, “Information sharing among multiple robots for cooperation in cellular robotic system,” in Intelligent Robots and Systems, 1997.

[19] G. Steinbauer, M. Faschinger, G. Fraser, A. Muhlenfeld, S. Richter, G. Wober, and J. Wolf, “Mostly harmless team description,” Graz University of Technology, Tech. Rep., 2003.

[20] A. Pahliani and P. Lima, “Cooperative opinion pool: a new method for sensor fusion by a robot team,” in Intelligent Robots and Systems, 2007.

[21] G. G. Matt Rosencrantz and S. Thrun, “Decentralized sensor fusion with distributed particle filters,” in Conference on Uncertatinty in AI (UAI), 2003.

[22] B. Upcroft, L. Ong, S. Kumar, M. Ridley, T. Bailey, S. Sukkarieh, and H. Durrant-Whyte, “Rich probabilistic representation for bearing only decentralized data fusion,” in 7th International Conference on Information Fusion, 2005.

[23] J. Messias, J. Santos, J. Estilita, and P. Lima, “Monte carlo localization based on gyrodometry and line-detection,” in Robótica 2008, 2008.

[24] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte carlo localization for mobile robots,” in IEEE International Conference on Robotics and Automation (ICRA99), 1999.

[25] P. Heinemann, J. Haase, and A. Zell, “A combined monte-carlo localization and tracking algorithm for robocup,” in International Conference on Intelligent Robots and Systems, 2006.

[26] S. Thrun, W. Bugard, and D. Fox, Probabilistic Robotics. The MIT Press, 2005.

[27] M. Taiana, J. Santos, J. Gaspar, J. Nascimento, A. Bernardino, and P. Lima, “Color 3d model-based tracking with arbitrary projection model,” in SIMPAR Omnidirectional Vision Workshop, 2008.

[28] A. Bhattacharyya, “On a measure of divergence between two statistical populations defined by their probability distributions,” in Bulletin Calcutta Mathematical Society, 1943.

[29] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. Springer, 2003, pp. 165–190, 236–242.

[30] J. A. Bilmes, “A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models,” U.C. Berkeley, Tech. Rep., 1998.

[31] K. V. Mardia, Multivariate Analysis. Academic Press, 1979, pp. 96–97.

[32] S. Julier and J. Uhlmann, “A non-divergent estimation algorithm in the presence of unknown correlations,” in American Control Conference, 1997.

[33] S. Kullback, Information Theory and Statistics. Dover Books, 1968.

[34] H. Beiji, S. Maes, and J. Sorensen, “A distance measure between collections of distributions and its application to speaker recognition,” in International Conference on Acoustics, Speech and Signal Processing, vol. 2, 1998, pp. 753–756.

[35] A. T. Ihler, J. W. F. III, and A. S. Willsky, “Particle filtering under communications constraints,” in Proceedings, IEEE Statistical Signal Processing (SSP), 2005.