Multi-Robot Cooperative Object Localization
Decentralized Bayesian Approach

João Santos

Institute for Systems and Robotics, IST, 1049-001 Lisboa, Portugal
jsantos@isr.ist.utl.pt

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. 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 will first implement 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 will then take the decentralized Bayesian approach to propose a cooperative sensor model that improves ball tracking and self-localization.

1 Introduction

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.

A sensor model describes the uncertainty associated with each sensor observation allowing to extract relevant information, rather than 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).

ISocRob 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.

In Section 2 we describe the implementation of a shape-based 3D tracker for the ball. Then in Section 3 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 4 we present several experimental results to validate the introduced methods. Section 5 outlines our conclusions.

2 Ball Detection and Tracking

The ball identification in the image is based on Taiana [1] ball projection model. A 3D model of the ball is used to calculate it’s 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 3D contour lies on the intersection with an orthogonal plane to the line connecting the projection center to the center of the sphere. Given a 3-dimensional position, the 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 $x_t = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$. 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, $p(x_t|x_{t-1})$, and that for every time step $t > 1$ a new observation $z_t$ about the state $x_t$ is made. Given the observation history at time $t$ by $Z_t = [z_1, ..., z_t]$ our goal is to estimate the posterior distribution $p(x_t|Z_t)$ for each time step. This can be done recursively over Prediction and Update:

$$p(x_t|Z_{t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|Z_{t-1})dx_{t-1}$$

$$p(x_t|Z_t) \propto p(z_t|x_t)p(x_t|Z_{t-1})$$

where $p(x_{t-1}|Z_{t-1})$ is the previous estimate and $p(z_t|x_t)$ 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 $N$ weighted samples $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$, such that the posterior is approximated by an empirical estimate:

$$p(x_t|Z_t) \approx \frac{1}{N} \sum_{i=1}^N w_t^{(i)} \delta(x_t - x_t^{(i)})$$

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

$$\hat{x} = \frac{1}{N} \sum_{i=1}^N w_t^{(i)} x_t^{(i)}$$
Prediction computes an approximation of $p(x_t | Z_{t-1})$ 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:

$$X_t = \begin{bmatrix} I \frac{(\Delta t)^2}{2} \\ 0 \frac{(\Delta t)}{2} I \end{bmatrix} X_{t-1} + \begin{bmatrix} \frac{(\Delta t)^2}{2} \\ \frac{(\Delta t)}{2} \end{bmatrix} \alpha_t$$

where $I$ is the $3 \times 3$ identity matrix, $\Delta t = 1$, and $\alpha_t$ is a $3 \times 1$ 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 consider its kinematic configuration and 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. We take 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 $x_r^t$. We compute the particles state given by the reference coordinate transformation yield by the robot’s motion, here modeled as a Gaussian with mean $\bar{u} = [\delta_x, \delta_y, \delta_\theta]^T$ and covariance matrix $\Sigma$, as

$$x_t^r = T_{rot} x_{t-1} + T_{shift} \bar{u}_{t-1}$$

where

$$T_{rot} = \begin{bmatrix} R_p & 0 \\ 0 & R_v^{-1} \end{bmatrix}, T_{shift} = [S \ 0]^T.$$

and $R_p$ and $R_v^{-1}$ 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

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

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

In the Update step, the particle’s weights are updated according to the computed likelihood $p(z_t | x_t^{(i)})$ for each hypothesis, from the observation model. We follow Taina’s [1] 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 [2] 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 [3].

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 constraint the detection according to the camera resolution.

3 Cooperative Perception in Mobile Sensor Networks

3.1 Information Representation

Non parametric discrete approximations, such as the ones described in Section 2, 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 [4]. If the kernel function of the mixture model is Gaussian, the distribution is expressed as a Gaussian Mixture Model (GMM) of the form:

\[ P(x) = \sum_{k=1}^{N} w_k G(x|\mu_k, \Sigma_k) \]  

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 \( \mu_k \) and covariance \( \Sigma_k \), and \( N \) is the total number of mixture components. 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.

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 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 (E-step) and a maximization step (M-step). In the E-step we compute the expected likelihood for the complete data \( \Gamma \) (also known as Q-function) as the conditional distribution of the missing data \( Y \), given the current settings of parameters \( \Theta \) and the observed incomplete data \( X \). So, using Bayes’s rule, for each mixture component \( k \):

\[
p(y_i = k|x_i, \theta_k) = \frac{p(y_i = k, x_i|\theta_k)}{p(x_i|\theta_k)} = \frac{p(x_i|y_i = k, \theta_k)p(y_i = k|\theta_k)}{\sum_{k=1}^{N} p(x_i|y_i = k, \theta_k)p(y_i = k|\theta_k)}
\]  

(10)
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. ?? 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. In the M-step we re-estimate the mixtures parameters \( \Theta \) by maximizing the Q-function, see [4],[5] for the in-depth derivation.

From here we can compute \( \Theta' \) for each component \( k \):

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

(11)

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

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

(12)

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:

\[
L(\Theta) = \ln p(X|\Theta) = \ln \sum_{k=1}^{N} p(x_i|y_i = k, \theta_k)p(y_i = k|\theta_k)
\]

(13)

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

\[
L(\Theta') - L(\Theta) = \ln(\sum_{k=1}^{N} p(x_i|y_i = k, \theta_k)w_k) - \ln p(X|\Theta)
\]

(14)

We assume convergence if \( L(\Theta') - L(\Theta) < \psi \) for a given threshold \( \psi \).

### 3.2 Cooperative Sensor Model

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. 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 perception framework in Section 2, 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. 1.

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

In the Team Filter we receive GMM representations of the ball’s posterior in the world frame. Regarding information fusion, the Covariance Intersection (CI) filter yields consistent estimates to the problem of combining different Gaussian random vectors. This can be extended to a GMM Covariance Intersection algorithm as in [6], by performing CI between each of the mixture components. The fusion between the \(i\)th component of a GMM and the \(j\)th component of another GMM will result in a Gaussian mixture with \(N \times N\) components, such that:

\[
\Sigma^{-1}_{ij} = \gamma \Sigma^{-1}_i + (1 - \gamma) \Sigma^{-1}_j \tag{15}
\]

\[
\mu_{ij} = \Sigma_{ij} (\gamma \Sigma^{-1}_i \mu_i + (1 - \gamma) \Sigma^{-1}_j \mu_j) \tag{16}
\]

\[
w_{ij} = \frac{1}{N} (\gamma w_i + (1 - \gamma) w_j) \tag{17}
\]

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 its observations. A distance measure between Gaussian distributions can be defined as Kullback-Leiber distance [7], Bhattacharyya distance [2] 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. [8] approach to measure distances between collections of distributions in speech recognition, and define our measure of divergence between GMMs as:

\[
D(G_1, G_2) = \sum_{i=1}^{N} \frac{w_i^1}{c_i} + \sum_{j=1}^{N} \frac{w_j^2}{c_j} \leq \xi \tag{18}
\]
and assume there is agreement if $D(G_1, G_2) \leq \xi$, where $\xi$ is a positive threshold. Consider the matrix of distances between $N \times N$ mixture componentes:

$$T = \begin{bmatrix}
d_{11} & d_{12} & \ldots & d_{1N} \\
d_{21} & d_{22} & \ldots & d_{2N} \\
\vdots & \vdots & \ddots & \vdots \\
d_{N1} & d_{N2} & \ldots & d_{NN}
\end{bmatrix} \quad (19)$$

$W^1_i$ is the minima of the elements in the row times the row number $c_i$. Likewise, $W^2_j$ 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 for multivariate Gaussian distributions.

### 3.3 Improving Self-Localization

The current self-localization method is based on the previous work of Messias, Santos, Estilita and Lima [9], in which we combine Monte Carlo Localization with gyrodometry and line points extraction. However, one of the issues that affects MCL performance is the ability to recover from failures. The normal approach 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. 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 $p(x_t|f_t^1, c^1_t, m)$, 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 $m_{1x}$ and $m_{1y}$ 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.

### 4 Experimental Results

#### 4.1 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. 2a, but other colored balls, e.g. orange, could and have been used [10]. Images on the robot where acquired using an omnidirectional camera with a dioptric setup at 10fps. Odometry motion control measurements were obtained at 25fps and we used only 600 particles in the tracker. The results are visible in Fig. 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).
4.2 Generating Compact Information Representations

In this experiment we tested the particle set approximation with EM. We ran our EM implementation with a different number of mixture components for the same scenario and registered the average run time of the algorithm in Table 1. All the processing was made online with 600 particles in the ball tracker and a minimum of 20 particles in the MCL.

Table 1. Average execution time while computing GMMs with our EM implementation for 12000 particles

<table>
<thead>
<tr>
<th>Number of mixture components</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time taken [seconds]</td>
<td>0.0246</td>
<td>0.0690</td>
<td>0.1131</td>
<td>0.1924</td>
</tr>
</tbody>
</table>

4.3 Fusing Data

In this experiment two robots are able to localize the ball, while a third robot (the goalkeeper) cannot (see figure 3). 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 CI. We set the threshold $\xi = 30$ from eq. ?? 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. 3f-g and represents the goalkeeper ball team estimate computed in the Team Filter.
5 Conclusions

We presented a cooperative sensor fusion model based on a particle filter perception framework, for mobile robots operating in dynamic environments. 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.

References

Fig. 3. GMM Data Fusion. Both Robot3 and Robot4 track the ball, computes its GMM and broadcast it. (a) Top field camera view. (b) Robot3 standard deviation ellipse of the computed modes and (c) approximation of the ball particle representation. (d) Robot 4 standard deviation ellipse of the computed modes and (e) approximation of the ball particle representation. (f) The goalkeeper (Robot1) tests received GMMs for disagreement and computes GMM CI. (g) The final fused GMM is consistent.