# Disambiguating Localization Symmetry through a Multi-Clustered Particle Filtering

Fabio Previtali[1] Guglielmo Gemignani[1] Luca Iocchi[1] Daniele Nardi[1]

***Abstract*— Distributed Particle filter-based algorithms have been proven effective tools to model non-linear and dynamic processes in Multi Robot Systems. In complex scenarios, where mobile agents are involved, it is crucial to disseminate reliable beliefs among agents to avoid the degradation of the global estimations. We present a cluster-based data association to boost the performance of a Distributed Particle Filter. Exploiting such data association, we propose a disambiguation method for the RoboCup scenario robust to noise and false perceptions. The results obtained using both a simulated and a real environment demonstrate the effectiveness of the proposed approach.**

## I. Introduction

Considering the new rules adopted in the RoboCup competition, Distributed Data Fusion will certainly become a breakthrough in the development and the realization of the next winning teams. In the RoboCup event, in fact, both the field size and the number of playing robots have been increased, making from now on localization, coordination and cooperation vital requirements to reach a good level of playing. Hence in this article a novel method will be presented to contribute to the development of an effective play for the future generation of winning teams.

Firstly, in order to tackle the distributed data fusion problem, we present a Distributed Particle Filtering (DPF) algorithm for tracking multiple objects. In particular, a team of agents run a two-tiered DPF to estimate the position of targets using the information shared among the teammates. A local Particle Filter (PF) estimates the targets' positions using the information gathered by a sensor with limited field-of-view. The data fusion rule utilizes the shared information as *Gaussian Mixture Models* (*GMM*), followed by two types of post-processing applied to the resulting particle cloud: *clusterization* and *data correlation*.

On top of the proposed Data Fusion method we built a disambiguation technique, robust to noise and false perceptions, that is based on a voting system. With this work we aim at contributing to the definitive resolution of the critical disambiguation problem in the RoboCup environment, as well as in more general ones.

## II. Related Work

The present work is based on the assumption that a team of robots performs a distributed position estimation of one or more targets in order to solve the localization symmetry problem. This assumption is intrinsically subdivided into two aspects: the distributed data fusion aspect and the symmetry disambiguation one.

The problem of multiple target tracking has been addressed by a vast amount of researchers. In this multifaceted and variegated community different solutions have been proposed, each specializing the used approach in the chosen application field. Such specialization has generated different algorithms, each having peculiar characteristics such as the usage of a clustering technique, synchronous or asynchronous features, centralized or decentralized computation, the exploitation of a compact information structure and so on. Out of this wide panorama we cite three remarkable works that have significantly contributed to the improvement of one or more aspects of multiple target tracking.

A notable work that describes a technique used to reduce the communication overhead is the article published by Gu [1], which presents a synchronous algorithm based on the exchange of GMM parameters that are evaluated using a distributed *Expectation-Maximization* (*EM*) method. Given this global approximation, particles and weights are extracted and propagated locally to the sensor nodes. An exemplifying approach that clusterizes the particle cloud in order to detect the targets' poses is instead given by Wu and Tong [2]. This article describes a *Boosted Interactively Distributed Particle Filter* (*B-IDPF*) that uses the boosting technique to initialize a variable number of tracking targets. The identities of the various targets are kept by the B-IDPF, allowing for a distinction of overlapping targets. In this work however, the computation is not distributed among several nodes, making the system less reliable and stable. Finally, an asynchronous approach is presented by Oreshkin and Coates [3], where the posterior probability is evaluated using a gossiping protocol for data dissemination. While this approach grants the advantages of asynchronous approaches, it increases however the communication overhead not exploiting the GMM technique.

Having these three works in mind, we proposed a filter that exploits a particular aspect of each of the cited works. In particular, a GMM technique is used to lower the overall communication overhead, a modified clusterization algorithm is exploited to track a number of targets not known a priori and, finally, an asynchronous approach is adopted to avoid the synchronization problems.

Regarding the different disambiguation methods proposed so far, two major different approaches have been presented. In 2012 the rUNSWift team [4] described an algorithm that exploits the background noise of two images taken at the beginning of the game, placing a robot in the center of the field that alternatively faces the two goal posts. The

[1]Authors are with the Department of Computer, Control, and Management Engineering, Sapienza University of Rome, via Ariosto 25, 00185, Rome, Italy

other method proposed compares instead the different robot models of the field based on the ball position and the robots' world beliefs. In particular, the Austin-Villa team [5] used a simple counter to keep track of the different ball estimations and to decide the possible re-localization actuation, while the B-Human team [6] exploited a fused ball model to check every robot field representation.

Our approach to the symmetry disambiguation problem is developed on top of the proposed distributed particle filter, adopting a model comparison technique. In particular, we built a symmetry disambiguation method robust to system perturbations and false perceptions that adopts a voting approach. Even though the model comparison technique has been previously proposed, exploiting the novel distributed particle filter enables us to take more reliable decisions, taking into account multiple variables (e.g., the different reliabilities of the sources) and being able to filter most of the input disturbs.

## III. Distributed Multi-Sensor Multiple Object Tracking

### A. Problem Definition

The Distributed Multi-Agent Multiple Object Tracking problem can be formalized as follows. Let $\mathcal{O} = \{o_1, \ldots, o_n\}$ be the set of all the moving objects and $\mathcal{S} = \{s_1, \ldots, s_S\}$ be the set of moving agents (also addressed as robots), each one having limited knowledge about the environment (e.g., 60.9° horizontal, 47.6° vertical in the experiments presented). The number $n$ of moving objects is unknown and can change over time. The set of measurements about the objects in the field-of-view of a sensor $s \in \mathcal{S}$ at a time $t$ is denoted by $z_{s,t} = \{z_{s,t}^{(1)}, \ldots, z_{s,t}^{(l)}\}$, where a measurement $z_{s,t}^{(i)}$ can be either a real object present in the environment or a false positive. The set of all the measurements gathered by all sensors at time $t$ is denoted by $z_{\mathcal{S},t} = \{z_{s,t} \,|\, s \in \mathcal{S}\}$. The history in time of all the measurements coming from all sensors is defined as $z_{\mathcal{S},1:t} = \{z_{\mathcal{S},j} : 1 \leq j \leq t\}$. It is worth noticing that, we do not assume the measurements generated by the sensors to be synchronized. The goal is to determine, for each sensor $s$, an estimation $x_{s,t}$ of the position of the objects at time $t$ in a distributed fashion.

### B. Distributed Multi-Clustered Particle Filtering

In order to achieve this goal, we estimate, for each sensor $s$, the position $x_{s,t} = \{x_{s,t}^{(1)}, \ldots, x_{s,t}^{(v)}\}$ of the objects by merging all the available information. Although the sensors continuously send information about their observations, the estimation computed by one sensor may be different from the others due to noise or delay in communication. Specifically, the overall objective is to determine the likelihood $p(x_{s,t} \,|\, z_{\mathcal{S},1:t})$ of the global estimation $x_{s,t}$ for each sensor $s$, given the observations $z_{\mathcal{S},1:t}$ collected by all sensors.

We assume that the acquired observations are affected by an unknown noise that is conditionally independent among the sensors. During the acquisition process, each sensor does not interact with the others, thus allowing for a factorization of the likelihood of the global estimation that can be expressed by the following joint likelihood:

$$p(z_{\mathcal{S},t}|x_{s,t}) = \prod_{s \in \mathcal{S}} p(z_{s,t}|x_{s,t}) \tag{1}$$

Given the assumption in Eq. (1), a fusion algorithm can be described using Bayesian Recursive Estimation:

$$p(x_{s,t}|z_{\mathcal{S},1:t}) = \frac{p(z_{\mathcal{S},t}|x_{s,t})p(x_{s,t}|z_{\mathcal{S},1:t-1})}{\int p(z_{\mathcal{S},t}|x_{s,t})p(x_{s,t}|z_{\mathcal{S},1:t-1})dx_{s,t}} \tag{2}$$

$$p(x_{s,t}|z_{\mathcal{S},1:t-1}) = \int p(x_{s,t}|x_{s,t-1})p(x_{s,t-1}|z_{\mathcal{S},1:t-1})dx_{s,t-1} \tag{3}$$

Eq. (2) and (3) represent a global recursive update that can be computed if and only if complete knowledge about the environment is available. Therefore, we propose to approximate the exact optimal Bayesian computation - Eq. (2) and 3) - by using a Distributed Particle Filter-based algorithm.

To this end, we devise a novel method, called *PTracking*, based on Distributed Multi-Clustered Particle Filtering. The algorithm is divided into two phases, namely a *local estimation* phase and a *global estimation* phase (Algorithm 1). Each sensor performs the local and global computation, sharing the obtained results in order to achieve a better representation of the current scene. The output is an estimation of the current state $x_{s,t} = (\boldsymbol{I}_{s,t}, \boldsymbol{\Lambda}_{s,t}, \boldsymbol{M}_{s,t}, \boldsymbol{\Sigma}_{s,t})$ having considered the previous state $x_{s,t-1}$ and the observations $z_{\mathcal{S},t}$. Where, $\boldsymbol{I}_{s,t}$ represents the set of global identities of sensor $s$, $\boldsymbol{\Lambda}_{s,t}$ the weights of the estimations, $\boldsymbol{M}_{s,t}$ contains the Cartesian position in the environment of the estimations while $\boldsymbol{\Sigma}_{s,t}$ describes the uncertainty about estimations.

**Local estimation.** The local estimation phase (Algorithm 1, lines 1-7) contains three steps: 1) A particle filtering step, that computes the evolution of the local estimations given the local observations $z_{s,t}$ provided by the sensor; 2) A clustering step that determines the *Gaussian Mixture Model* (*GMM*) parameters of this distribution; 3) A *data association* step to assign an identity to each object $o \in \mathcal{O}$.

The prediction step of the PF uses an initial guessed distribution, based on a *transition state* model $\pi$. Such a transition model makes a prediction of the next state based on the sensor movement. Then, using the previously computed state $x_{s,t-1}$, the transition model, given by the measurements $z_{s,t}$, is applied. Afterwards, from this hypothesized distribution, a set of samples is drawn and weighted exploiting the current local perception $z_{s,t}$. Finally, the *Sampling Importance Resampling* (*SIR*) principle is used to re-sample the particles which are then clustered in order to determine the parameters of the final GMM model. It is worth noticing that, in contrast to other related approaches, this step enables the creation of a more compact information structure allowing us to drastically reduce the communication overhead. A *data association* step is then applied to assign an identity (track number) to each object.

When the final GMM set has been computed, each sensor broadcasts the set of GMM parameters describing all the objects detected.

**KClusterize.** The clustering phase is performed by using a novel clustering algorithm, called *KClusterize*, aiming

**Algorithm 1:** PTracking

**Input**: perceptions $z_{s,t}$, local track numbers $i_{s,t-1}$, global track numbers $I_{s,t-1}$

**Data**: set of local particles $\tilde{\xi}_{s,t}$, set of global particles $\tilde{\xi}_{\mathcal{S}',t}$, sensor pose $m_{s,t}$, local GMM set $\mathcal{L}$, global GMM set $\mathcal{G}$

**Output**: global estimations $x_{s,t} = (\boldsymbol{I}_{s,t}, \boldsymbol{\Lambda}_{s,t}, \boldsymbol{M}_{s,t}, \boldsymbol{\Sigma}_{s,t})$

1 **begin**
2 $\quad \tilde{\xi}_{s,t} \sim \pi_t(x_{s,t}|x_{s,t-1}, z_{s,t}, m_{s,t})$
3 $\quad$ Re-sample by using the SIR principle
4 $\quad \mathcal{L} = KClusterize(\tilde{\xi}_{s,t})$
5 $\quad (\boldsymbol{i}_{s,t}, \boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t}) = DataAssociation(\mathcal{L}, i_{s,t-1})$
6 $\quad$ Communicate belief $(\boldsymbol{i}_{s,t}, \boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$ to other agents
7 **end**
8 **begin**
9 $\quad$ Collect $\mathcal{L}_{S'}$ from a subset $\mathcal{S}' \subseteq \mathcal{S}$ of sensors within a $\Delta t$
10 $\quad \tilde{\xi}_{\mathcal{S}',t} \sim \tilde{\pi} = \sum_{s \in \mathcal{S}'} \boldsymbol{\lambda}_{s,t} \mathcal{N}(\boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$
11 $\quad$ Re-sample by using the SIR principle
12 $\quad \mathcal{G} = KClusterize(\tilde{\xi}_{\mathcal{S}',t})$
13 $\quad (\boldsymbol{I}_{s,t}, \boldsymbol{\Lambda}_{s,t}, \boldsymbol{M}_{s,t}, \boldsymbol{\Sigma}_{s,t}) = DataAssociation(\mathcal{G}, I_{s,t-1})$
14 **end**

at fulfilling the following three requirements: 1) number of objects to be detected cannot be known a priori, 2) low computational load for real-time applications and 3) Gaussian distribution for each cluster.

Alternative clustering methods are not adequate since they either need to know the number of clusters in advance (e.g., *k-means*), or they are computationally expensive and not real-time (e.g., free-clustering algorithms like *Expectation-Maximization* [7], *BSAS* [8] or *QT-Clustering* [9]). KClusterize does not require any initialization, it has a linear complexity and all the obtained clusters reflect a Gaussian distribution.

More specifically, KClusterize first clusters the particles trying to find all the possible Gaussian distributions. Then, a post-processing step is applied to verify that each cluster actually represents a Gaussian distribution. To this end, all the non-Gaussian clusters are split (if possible) into Gaussian clusters. It is worth noticing that, the final number of Gaussian distribution components provided as output can be different from the one found during the first step. Finally, using such clusters a GMM set $(\boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$, representing the estimations performed by the sensor $s$, is created.

**Global estimation.** The global estimation phase (Algorithm 1, lines 8-14) starts receiving information from other sensors. Notice that, as already mentioned, the proposed method is asynchronous and the collection of information is limited to a small amount of time $\Delta t$. During this time information is received from a subset $\mathcal{S}' \subseteq \mathcal{S}$ of sensors. This mechanism is thus robust to communication delays and dead nodes, since the global estimation phase proceeds even if some node is not communicating or the communication channel is not reliable. Once data have been gathered, a particle set $\tilde{\xi}_{\mathcal{S}',t}$ is updated using the received GMM parameters $(\boldsymbol{i}_{s,t}, \boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$ for $s \in \mathcal{S}'$. These particles are re-sampled in order to extract reliable quality information about the global estimates. Then, a weighting procedure is applied to the set. Instead of weighting the particles by using the whole pool of GMM parameters, we cluster them by again using *KClusterize* to obtain a new GMM pool. The weighting of particles is performed using such a new GMM pool. In this way the assigned weights are more consistent since only the most relevant parameters are considered. The global estimation phase determines the GMM parameter set of the tracked objects considering all the information available at time $t$ (local observations and information received by other sensors). Finally, a *data association* step is applied to assign an identity to each object considering all the available information received by other sensors.

**Algorithm 2:** KClusterize

**Input**: particle set $\mathcal{P} = \{p_1, \ldots, p_m\}$

**Data**: set of centroids $\mathcal{F}$, cluster of particles $c_i$, sets of Gaussian clusters $\mathcal{Q}$ and $\mathcal{C}$

**Output**: GMM set $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\sigma})$

1 $initialize\ \mathcal{F} = \varnothing$
2 **foreach** $p_i \in \mathcal{P}$ **do**
3 $\quad$ **if** $\forall f_k \in \mathcal{F}\ \{\|p_i, f_k\| > \delta_{model}\}$ **then**
4 $\quad\quad \mathcal{F} = \mathcal{F} \cup \{p_i\}$
5 $\quad$ **end**
6 **end**
7 $c_i = \varnothing \quad \forall i \in [1, |\mathcal{F}|]$
8 **foreach** $p_i \in \mathcal{P}$ **do**
9 $\quad$ **foreach** $f_k \in \mathcal{F}$ **do**
10 $\quad\quad$ **if** $\|p_i, f_k\| < \delta_{model}$ **then**
11 $\quad\quad\quad c_k = c_k \cup \{p_i\}$
12 $\quad\quad$ **end**
13 $\quad$ **end**
14 **end**
15 $initialize\ \mathcal{C} = \varnothing$
16 **foreach** $c_i$ **do**
17 $\quad$ **if** $c_i \not\sim \mathcal{N}(\mu, \sigma)$ **then**
18 $\quad\quad \mathcal{Q} = KClusterize(c_i)$
19 $\quad\quad$ **foreach** $q_j \in \mathcal{Q}$ **do**
20 $\quad\quad\quad$ **if** $q_j \sim \mathcal{N}(\mu, \sigma)$ **then**
21 $\quad\quad\quad\quad \mathcal{C} = \mathcal{C} \cup \{q_j\}$
22 $\quad\quad\quad$ **end**
23 $\quad\quad$ **end**
24 $\quad$ **end**
25 **end**
26 *compute* $(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ *from* $\mathcal{C}$

**Data association.** An identity (i.e., a track number) has to be assigned to each object, by associating the new observations to the existing tracks. This is the most difficult and fundamental step for any tracking algorithm. In our approach, we consider as features for the data association the direction, the velocity and the position of the objects. Complete and partial occlusions can occur when objects are aligned with respect to the sensor view or when they are close to each other, making visual tracking hard. Our solution is to consider collapsing tracks to form a group, instead of tracking them separately. When two or more tracks have boundaries moving closer to each other, the tracker

saves their models and it merges them into a group that evolves considering both the estimated trajectory and the observations coming from the detector. When an occluded object becomes visible again, the stored models are used to re-assign the correct identification number, belonging to the corresponding previously registered track.

## IV. Field Symmetry Disambiguator

A critical problem present in the majority, if not all, of the localization algorithms used in the RoboCup scenario is represented by the symmetry disambiguation challenge. Being the soccer field symmetric it is really hard to unequivocally distinguish the two different playing sides. In the past, in order to facilitate the players, each goal post was colored differently from the opposite one. In 2012, the RoboCup committee decided to paint yellow both the goal posts in order to encourage the research community to solve the disambiguation problem. Having this problem in mind, we analyzed different possible solutions based on the distributed data fusion approach described in the previous section.

Since nowadays the localization algorithms are reliable and stable, we have noticed that the majority of the team players are most of the time well localized. Starting from this observation, we have developed a disambiguation method based on a voting system that relies on the previously presented algorithm. In order to describe this method, we will use a reference frame that has its origin placed at the center of the field, the x axis orientated toward the opponent goal and a counter-clockwise y axis.

Being each player able to give an estimation of its position on the field as well as the ball relative position, it is possible to guess the global ball position and compare it with the ones given by the data fusion algorithm. If the signs of the own ball coordinates are inverted - i.e., *signsDiff* returns true - with respect to the one given by the data fusion algorithm while their module is similar - i.e., *modulesSim* returns true, the robot can understand that it is inversely localized and start a re-localization process. In pseudo-code:

```
if (signsDiff(ownBallPos,fusedBallPos) &&
   (modulesSim(ownBallPos,fusedBallPos)) {
   ... \* Invert Signs *\
}
```

Analyzing this idea two main problems can be identified: 1) the algorithm will not guarantee a correct disambiguation when the majority of the teammates are not correctly localized (i.e., it depends on the ball estimation confidence of each unlocalized robot), 2) the algorithm will not be reliable when the ball is close to the field lines of symmetry. In fact, being the global ball coordinates close to zero, the presence of a natural error given by the image processor will cause fluctuations of the estimations signs, resulting in a system instability. While the first problem can be partially addressed by using different weights on the players, the second one can be overcome for the majority of the cases. In order to do so, firstly, we have divided the field in four different zones as shown in Fig. 1.

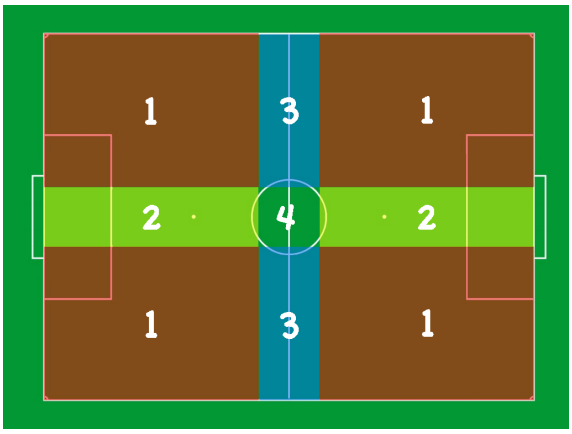


Fig. 1. Subdivision of the soccer field for the proposed disambiguation method. The dimensions of the zones depend on the error introduced by the vision algorithm in estimating the ball position, thus cannot be determined a priori.

Such zones have been chosen using a threshold distance from the two field symmetry lines. Such threshold distance cannot be decided a priori but it needs to be chosen depending on the error introduced by the vision algorithm during the estimation of the ball position.

Having divided the field in such zones, thus decomposing the problem in four sub-problems, we have analyzed each resulting case. In the first case, where the ball is located in the zones labeled with 1, the algorithm is not affected by the fluctuation problem and so it does not require a modification. In the second case (label 2), the algorithm can be modified considering only the y coordinate which does not fluctuate, resulting in:

```
if (YSignsDiff(ownBallPos,fusedBallPos) &&
   (modulesSim(ownBallPos,fusedBallPos)) {
   ... \* Invert Y Sign *\
}
```

We can treat the third case (label 3) in an analogous way, switching the x and y coordinates. The last case (label 4), instead, represents the only case in which the algorithm does not work. In fact, being the ball too close to the origin, the signs of both coordinates fluctuate, making the whole system unstable. While the flaws of the algorithm cannot be corrected in this very limited area, yet instability zone can also be shrunk reducing the image processor error. Moreover, this area represents the least important case for disambiguation, being far from the ends of the field; thus, a localization error in this area may have a low impact on the game.

## V. Experiments

In order to evaluate the performance of our work, we tested the system both in a simulated environment and in a real one. Since these two environments are correlated to the *RoboCup Standard Platform League*, we used a ball as the target to be tracked by three *Aldebaran Nao* robots. The experiments have been carried out using the B-Human code release 2012 [6] as a baseline, developing our code upon it. In particular, we used their vision and localization modules in addition

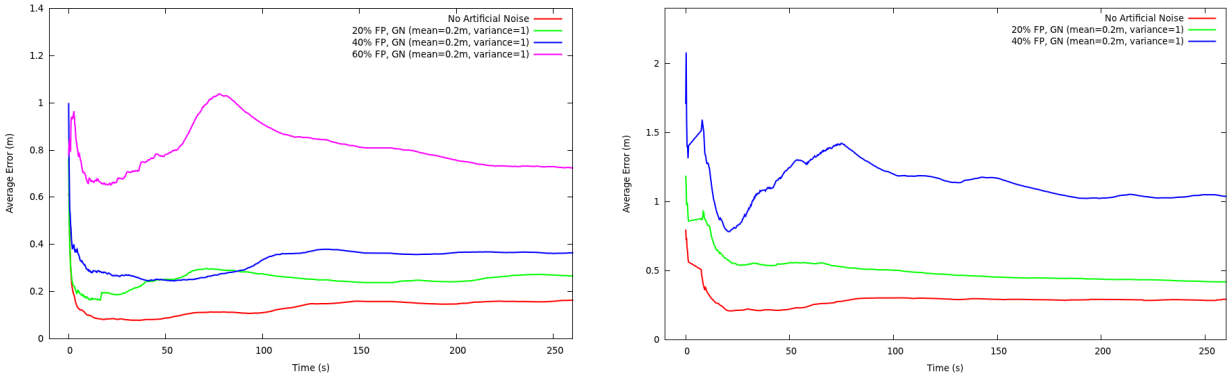


Fig. 2. The left picture shows the trend of the global ball estimation error with respect to time using the robot pose ground-truth, while the right plot illustrates the same evaluation using the robot localization algorithm.

to the SimRobot [10] simulator, where we tested the whole system.

### A. SimRobot

**Experiment definition.** The objective of the experiments is to show the robustness of the tracking system to false positives, the estimations' accuracy of the global ball position on the field and the benefits obtained from the proposed distributed data fusion technique. These experiments consisted in having three simulated Aldebaran Nao robots in motion tracking the ball movements on the field. During such tests, an artificial noise has been incrementally added to the perceptions in order to show the robustness of the tracking system. In particular, bursts of 20%, 40% and 60% of false perceptions per second and a Gaussian noise with $\mu = 0.2m$ and $\sigma = 1$ have been added. Each experiment has been repeated three times with a duration of five minutes per run. Moreover, for the disambiguation technique, the robot pose ground-truth has been used to force an agent to be inversely localized (while the other two were correctly localized) and to monitor the correct result of the algorithm.

**Metrics.** The metric used to evaluate the estimation accuracy is the *Euclidean distance* between the estimated global ball position $e_t \doteq (x_t^e, y_t^e)$ at time $t$ and the real global ball position $r_t \doteq (x_t^r, y_t^r)$ provided by the ground-truth at the same time $t$:

$$EstimationError(t) = \sqrt{(x_t^e - x_t^r)^2 + (y_t^e - y_t^r)^2}$$

**Results.** The error made by the vision algorithm in estimating the ball position as well as the error made by the localization algorithm in estimating the robot pose in the field are respectively $0.15 \pm 0.03m$ and $0.5 \pm 0.1m$. These errors have been calculated using the Euclidean distance between the estimated position provided by the algorithm and the real position provided by the ground-truth.

The comparison between the global ball estimation accuracy, obtained by using the robot pose ground-truth, and the robot localization algorithm is shown in Fig. 2. Both measurements depend on the artificial perception noise. These results indicate that the proposed MCPF algorithm reaches a good accuracy in the estimation of the global ball position. Indeed, the left plot in Fig. 2 shows that the error made by the MCPF is around 15 cm that equals the error of the vision algorithm. Therefore, for this experiment we can derive that the designed method has not introduced an additional noise during the global ball estimation phase. Moreover, if we consider the outcomes when an artificial noise has been added, we can see that the MCPF algorithm is very robust to false perceptions. In fact, adding in input a Gaussian noise ($\mu = 0.2m$, $\sigma = 1$) and bursts of 20% of false positives per second, we have a global estimation error less than 30 cm and less than 50 cm respectively for the robots using the ground-truth and the one using the localization algorithm.

Fig. 3 shows instead a plot that highlights the robustness of the Distributed Data Fusion technique that has been developed. In fact, even if a robot has some localization problems, the proposed MCPF algorithm can handle them. In particular, in order to demonstrate so, the robot 1 (green line) was voluntary moved in different points of the field to create robot localization problems. These phenomena did not substantially affect the global ball estimation, even if bursts of 20% of false perceptions and a Gaussian noise were added in input (Fig. 3, plot on the right).
For the disambiguation part due to lack of space, a zipped folder, containing a video of the experiment and the related produced plots, has been uploaded on line[1]. In the video two robots, walking on the spot, alternatively signal their inverse localization. The related plots, in which the filtered robot position is colored in green, show a correct operation of the method.

### B. Real robot

**Experiment definition.** The objective of the experiments is to replicate in a real environment the tests made in the simulation environment, verifying the efficiency of the tracking system. The ground-truth of all the objects has been acquired measuring their fixed position on the field.

**Metrics.** As in the simulative case, the metric used to evaluate the estimation accuracy is the Euclidean distance between the estimated global ball position and the real global ball position.

**Results.** The measurements obtained in the experiment made with real robots confirmed the trend observed in the simulative environment. In fact, as shown in Fig. 4 the

[1] www.dis.uniroma1.it/~previtali/downloads/MFI-2015.zip

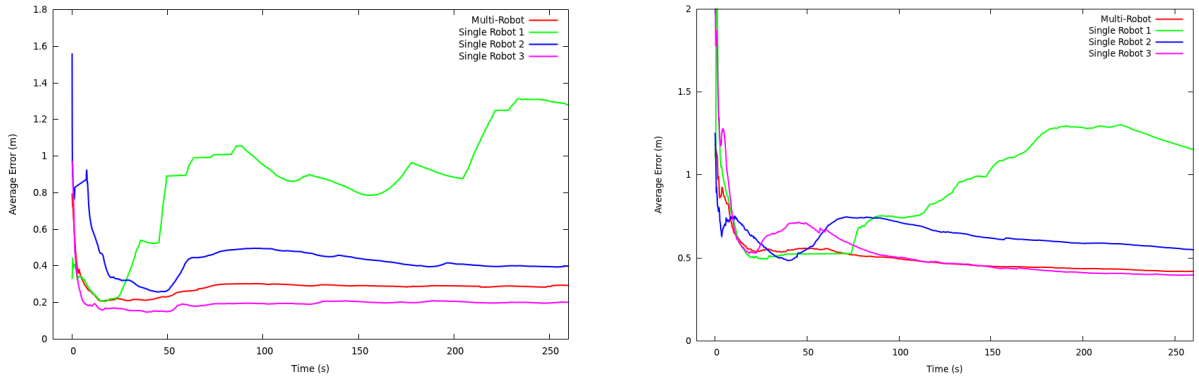


Fig. 3. The left plot illustrates the results of the experiment with no artificial noise while the right figure shows the case where a Gaussian noise and bursts of 20% of false perceptions were added to the perceptions. During these experiments, Robot 1 (green line) was voluntary moved in different positions in the field to create localization problems. It can be seen that these localization errors do not affect the multi-robot global ball estimation (red line).

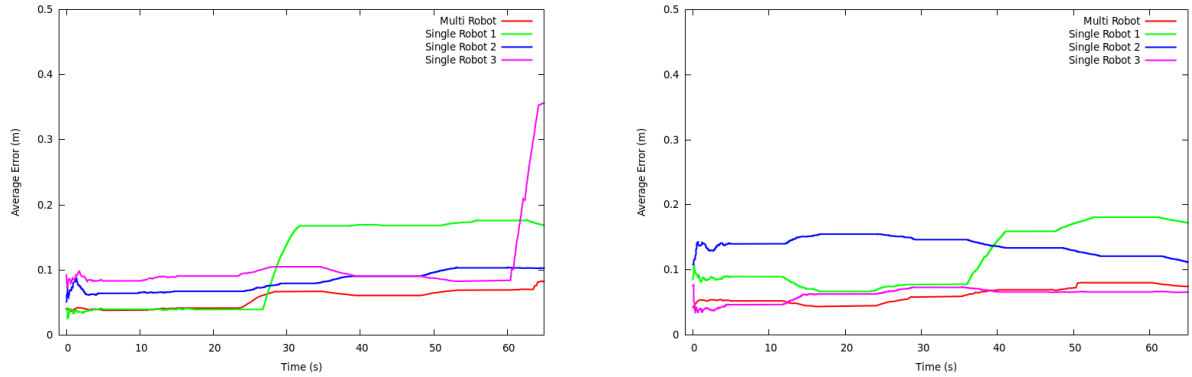


Fig. 4. The left plot illustrates the results of the experiment on real robots using the robot pose ground-truth while the right figure shows the case in which the robot localization algorithm was used.

proposed MCPF algorithm reaches a good accuracy in the estimation of the global ball position using both the robot pose ground-truth and the robot localization algorithm.

## VI. Conclusions

In order to tackle the distributed data fusion problem, we have presented a reliable distributed particle filtering algorithm for multiple objects tracking. Such an algorithm has been designed with a two-tiered structure: a local layer responsible of performing local estimations and a global layer utilized to fuse the measurements received by the teammates. Based on this data fusion method, a robust disambiguating technique for the localization symmetry has been described, showing a reliable method for the symmetry resolution in this specific RoboCup scenario. Specifically, a voting system that exploits the comparison between the local ball position and the one given by the data fusion algorithm has been described. Such technique enables us to take dependable decisions based on multiple variables due to the system robustness to noise and false positives.

This contribution has been tested both in a simulative and in a real environment and it has been proven to be an effective and reliable tool to tackle the considered problems. In particular, the data fusion technique has been demonstrated to be resilient to false perceptions and Gaussian noise, while the disambiguation method has been shown to be a reliable solution to partially solve the localization symmetry problem. In the future, we expect to integrate a distributed representation of various other targets on the field (e.g., goal posts, robots, etc.) that we think will lead to a complete resolution of the symmetry problem. Moreover we expect to perform further tests during the games of the next competitions, using a ground truth system to track the different robot and ball positions over time, thus comparing our algorithm performance with the other proposed in literature.

## References

[1] G. Dongbing, "Distributed particle filter for target tracking," in *IEEE ICRA*, 2007, pp. 3856–3861.

[2] Y. Wu, X. Tong, *et al.*, "Boosted interactively distributed particle filter for automatic multi object tracking," in *IEEE ICIP*, 2008.

[3] B. N. Oreshkin and M. J. Coates, "Asynchronous distributed particle filter via decentralized evaluation of gaussian products," in *International Conference on Information Fusion*, 2010, pp. 1–8.

[4] S. Harris, P. Anderson, *et al.*, "Robocup standard platform league - runswift 2012 innovations," in *Proceedings of Australasian Conference on Robotics and Automation*, 2012.

[5] S. Barrett, K. Genter, *et al.*, "Ut austin villa 2012: Standard platform league world champions," in *RoboCup-2012: Robot Soccer World Cup XVI*, 2013, pp. 129–136.

[6] T. Röfer, T. Laue, J. Müller, *et al.*, "B-human team report and code release 2012," 2012, only available online: www.b-human.de/downloads/coderelease2012.pdf.

[7] T. K. Moon, "The expectation-maximization algorithm," *Signal processing magazine, IEEE*, vol. 13, no. 6, pp. 47–60, 1996.

[8] S. Theodoridis and K. Koutroumbas, in *Pattern Recognition, Academic Press*, 2003, p. 433.

[9] L. J. Heyer, S. Kruglyak, and S. Yooseph, "Exploring expression data: identification and analysis of coexpressed genes," *Genome research*, vol. 9, no. 11, pp. 1106–1115, 1999.

[10] T. Röfer, T. Laue, J. Müller, *et al.*, "B-human team report and code release 2011," 2011, only available online: www.b-human.de/downloads/bhuman11_coderelease.pdf.