Robust Tracking of Multiple Soccer Robots using Random Finite Sets

Pablo Cano and Javier Ruiz-del-Solar

Advanced Mining Technology Center & Dept. of Elect. Eng., Universidad de Chile
{pcano,jruizd}@ing.uchile.cl

Abstract. Having a good estimation of the robot-players positions is becoming imperative to accomplish high level tasks in any RoboCup League. Classical approaches use a vector representation of the robot positions and Bayesian filters to propagate them over time. However, these approaches have data association problems in real game situations. In order to tackle this issue, this paper presents a new method for building robot maps using Random Finite Sets (RFS). The method is applied to the problem of estimating the position of the teammates and opponents in the SPL league. Considering the computational capabilities of Nao robots, the GM-PHD implementation of RFS is used. In this implementation, the estimations of the robot positions and the robot observations are represented using Mixture of Gaussians, but instead of associating a robot or an observation to a given Gaussian, the weight of each Gaussian maintains an estimation of the number of robots that it represents. The proposed method is validated in several real game situations and compared with a classical EKF based approach. The proposed GM-PHD method shows a much better performance, being able to deal with most of the data association problems, even being able to manage complex situations such as robot kidnappings.

Keywords: World Modeling, Multi-target tracking, Robot position estimation, Random Finite Sets

1 Introduction

As RoboCup progresses over the years, high-level skills become necessary to maintain a competitive level. Such skills are no longer restricted to the detection of field objects or to the self-localization of the robot players, but include team's skills based on the tracking (position estimation) of teammates and opponents. Examples of these skills are ball passing, adversaries' tracking and team's formation.

When the observability of the game and the players is not an issue to address, as in the case of the Small-size league, high- levels skills based on the tracking of the robot players have been already implemented (e.g. reactive coordination [1], and the analysis and learning of the opponent's strategies [2][3]).

The situation in the Standard Platform League (SPL) is more complex given the restricted field of view of the robot´s camera and the low computational resources of the Nao robots; in this league the detection of other robots is not robust and the construction of a good map of obstacles/players is a difficult process. Current standard robot tracking systems maintain/update the robot estimations using Kalman filters (e.g. [4][5]). However, in this tracking paradigm there is no clear solution of the data association problem, and several heuristics need to be used in order to eliminate and merge hypotheses.

In this context the main goal of this paper is to propose a new methodology for the robust tracking (position estimation) of multiple soccer robots using the Random Finite Sets (RFS) framework, which allows to overcome the drawbacks of current approaches. The proposed methodology is inspired in [6] where the term Probability Hypothesis Density (PHD) was introduced as the first moment of a point process. Then, the PHD filter is presented in [7] as a way to maintain hypotheses of multiple objects using sets instead of vectors to describe the object's states.

There are several works that use this new framework in the literature, concerning all kind of problems and subjects [8]. Principally, it is used in highly complex environments to track large amounts of features, which makes it very expensive computationally. But, in the SPL problem the number of features (robots) to detect rises to 10 in the worst case, which make it computationally tractable for a Nao robot's CPU.

The paper is organized as follows: in Section 2 the problem to resolve is described. Section 3 presents a brief introduction to RFS. Section 4 shows the implementation used in this work, and Section 5 the experimental results. Finally, conclusions are drawn in Section 6.

2 Problem Description: Data Association when Tracking Multiple Players in Robot Soccer

As already mentioned, knowing the position of the other robots in the field is relevant for implementing high-level soccer behaviors. In this work we will call *map of obstacles* to a map that a given player builds, and which includes the positions on the field of every other robot player, teammate or opponent (see Figure 1). We will denote observations to the detections of these robots, and obstacles to the estimated position of these robots in the map.

Most of the existing methods used for estimating the map of obstacles employ a classical approach with a vector representation of the obstacles (robots), which are propagated over time using a Bayesian filter (e.g. an EKF filter). However, it has been demonstrated that the use of a vector representation of the obstacles has numerous drawbacks, mainly related to the data association between new and past observations (obstacles) [9]. Some examples of those problems are illustrated in Figure 2: Fig. 2(a) shows a trivial case where the data association between two new observations (red crosses) and two obstacles (black crosses) is trivial. However, the data associations are not trivial in the cases illustrated in Figs. 2(b) and 2(c). First, Fig. 2(b) shows the case when two new measurements have a similar distance to the obstacle, in addition to be not very close to the obstacle (see the covariance of the obstacle representation). So, depending of the implemented data association strategy, this could end in one, two or three obstacles in the map. Fig. 2(c) show a case where two obstacles are very close, so the new measurement could be associated with any of them, and leave the other with no update for that frame. For these cases, most of methods use heuristics to associate new measurements to the obstacles, or to create new obstacles if no association is made. But, in a highly dynamic environment as a robot soccer match, these methods may produce several bad associations or missed detections.

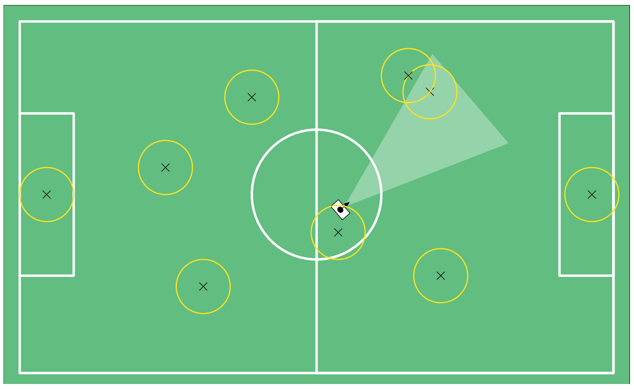


Fig. 1. Example of a map of obstacles. The white rectangle represents the robot which is building the map. The black crosses represent the robots/obstacles positions and the yellow circles the corresponding covariance of each representation. The lighted zone represents the Field of View of the camera.

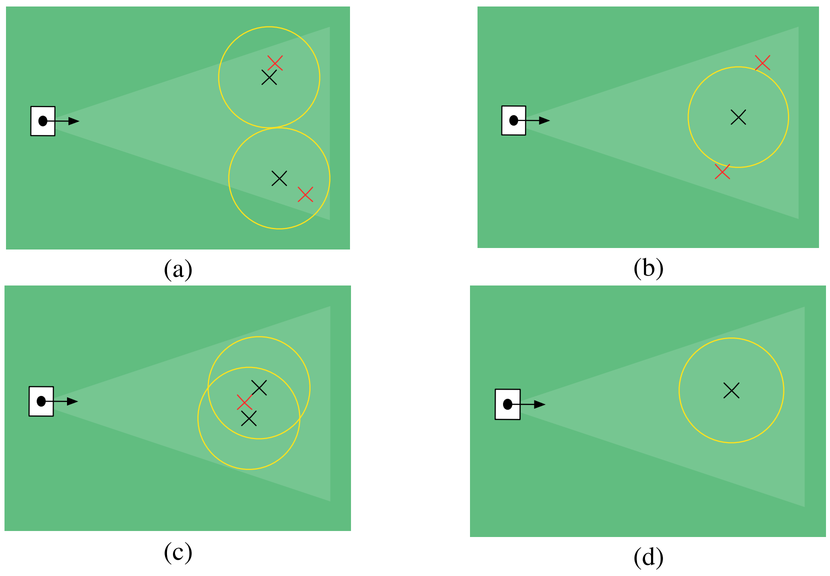


Fig. 2. The red crosses represent measurements of the sensor, black crosses represent the robots/obstacles positions and the yellow circles the corresponding covariance of each

representation. (a) represents an easy case of data association, while (b) and (c) show more complex cases. (d) represents a case when an obstacle that should be detected by the robot is not sensed.

Finally, Fig. 2 (d) describes a situation where no measurement is obtained for an obstacle inside the Field of View (FoV). For the classical approach, this is not different from an obstacle outside the FoV, and its only consequence is that the obstacles' covariance grows. So, depending of the speed of the covariance's growing (which maintain the obstacles outside the FoV), the obstacles inside the FoV will be maintained the same time that the others, although they do not receive any measurements.

3 Multi-target tracking with Random Finite Sets

The main idea of the proposed methodology is to use *finite sets* instead of vectors for representing both observations and obstacles, which can encapsulate positions and quantity uncertainty. As has been widely demonstrated [7][10][9], the first moment of RFS, known as Probability Hypothesis Density (PHD), can be used to construct a filter which propagates the PHD of the map posterior instead of the map posterior itself.

3.1 PHD Filter

The PHD function v at a point represents the density of the expected number of obstacles occurring at that point of the state space (map). Therefore, a property of the PHD is that for any given region S of the map

$$\mathbb{E}[|M \cap S|] = \int_S v(m)dm \tag{1}$$

where M represents the map RFS and $|\cdot|$ denotes the cardinality of a set. This means that, by integrating the PHD on any region S of the map, we obtain the expected number of obstacles in S [7].

The PHD filter considers the following two steps [7]:

- Prediction:

$$v_{k|k-1}(m) = v_{k-1}(m) + b_k(m) \tag{2}$$

where $b_k(m)$ represents the PHD of the new obstacles in time k.

- Update:

$$v_k(m) = v_{k|k-1}(m)\left[1 - P_D(m) + \sum_{z \in Z_k} \frac{P_D(m)g_k(z|m))}{c_k(z) + \int P_D(\xi)g_k(z|\xi)v_{k|k-1}(\xi)d\xi}\right] \tag{3}$$

where $\mathrm{P_D}(m)$ represents the probability of detecting an obstacle at m, $g_k(z|m)$ represents the likelihood that z is generated by an obstacle at m at time k (i.e. the measurement likelihood) and $c_k(z)$ is the clutter intensity at time k.

3.2 Considerations

To adopt this framework to the presented problem, some considerations must be done. As presented before, $P_D(m)$ represents the probability of detecting an obstacle at m, but it does not take into account the capability of the robot to sense at m. So the real probability is represented by $P_D(m|X_k)$ where X_k represent the position of the robot in time k. The same occurs with g_k, c_k and b_k, because they also depend of the robot position.

4 Implementation

There are many implementations of RFS in the literature, but most of them are time consuming, especially for Nao robots with limited computational capabilities [11][12]. Hence we use the Mixture of Gaussian implementation (GM-PHD) [13], because it is very time efficient and it allows to easily get the positions of the obstacles from the PHD.

4.1 GM-PHD implementation

The main idea is to represent any RFS as a mixture of Gaussians. Therefore, both obstacles and detections are represented by Gaussians. But, to represent the position and number uncertainty of the obstacles present in the field, it is necessary to add a weight to every Gaussian. In this way their positions represent the multitude of location of obstacles in the map while their weights represent the number of obstacles in that given region. So, a PHD map is a Gaussian Mixture of the form,

$$v_{k-1}(m|X_{k-1}) = \sum_{j=1}^{J_{k-1}} \omega_{k-1}^{(j)} \mathcal{N}\left(m; \mu_{k-1}^{(j)}, P_{k-1}^{(j)}\right) \quad (4)$$

which is a mixture of J_{k-1} Gaussians, with $\omega_{k-1}^{(j)}, \mu_{k-1}^{(j)}$ and $P_{k-1}^{(j)}$ being their corresponding prior weights, means and covariances, respectively. The same form is used to represent the new obstacles at time k, $b_k(m|Z_{k-1}, X_{k-1})$, as

$$b_k(m|Z_{k-1}, X_{k-1}) = \sum_{j-1}^{J_{b,k}} \omega_{k|k-1}^{(j)} \mathcal{N}\left(m; \mu_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}\right) \quad (5)$$

where $J_{b,k}$ is the number of Gaussians in the new PHD at time k, Z_{k-1} is the vector of measurements at time $k-1$ and $\omega_{k|k-1}^{(j)}, \mu_{k|k-1}^{(j)}$ and $P_{k|k-1}^{(j)}$ determine the shape of the PHD of new obstacles. Therefore, the predicted PHD of the map, shown in (2) is also a Gaussian mixture

$$v_{k|k-1}(m|X_k) = \sum_{j=1}^{J_{k|k-1}} \omega_{k|k-1}^{(j)} \mathcal{N}\left(m; \mu_{k|k-1}^{(j)}, P_{k|k-1}^{(j)}\right) \quad (6)$$

where $J_{k|k-1} = J_{k-1} + J_{b,k}$ are the number of Gaussians representing the union of the prior map PHD $v_{k-1}(m|X_{k-1})$, and the new obstacles PHD at time k. $\omega^{(j)}_{k|k-1}, \mu^{(j)}_{k|k-1}$ and $P^{(j)}_{k|k-1}$ represents the shape and form of the Gaussians of the prior map PHD if $j < J_{k-1}$ and the shape and form of the Gaussians of the new observations PHD otherwise.

So, the posterior PHD shown in (3) is also a Gaussian mixture of the form

$$v_k(m|X_k) = v_{k|k-1}(m|X_k)\left[1 - P_D(m|X_k) + \sum_{z\in Z_k}\sum_{j=1}^{J_{k|k-1}} v^{(j)}_{G,k}(z, m|X_k)\right] \tag{7}$$

where $v^{(j)}_{G,k}$ corresponds, according to the general PHD Filter update equation, to

$$v^{(j)}_{G,k}(z, m|X_k) = \omega^{(j)}_k(z|X_k)\mathcal{N}\left(m; \mu^{(j)}_{k|k}, P^{(j)}_{k|k}\right) \tag{8}$$

$$\omega^{(j)}_k(z|X_k) = \frac{P_D(m|X_k)\omega^{(j)}_{k|k-1}q^{(j)}(z|X_k)}{c_k(z) + \sum_{i=1}^{J_{k|k-1}} P_D(m|X_k)\,\omega^{(i)}_{k|k-1}q^{(i)}(z|X_k)} \tag{9}$$

where $q^{(i)}(z|X_k) = \mathcal{N}\left(z; H_k\mu^{(i)}_{k|k-1}, S_k\right)$ is the measurement likelihood. The components $\mu^{(i)}_{k|k}$ and $P^{(i)}_{k|k}$ can be obtained from the standard EKF update equations,

$$S^{(i)}_k = R_k + \nabla H_k P^{(i)}_{k|k}\nabla H^T_k \tag{10}$$

$$K^{(i)}_k = P^{(i)}_{k|k}\nabla H^T_k\left[S^{(i)}_k\right]^{-1} \tag{11}$$

$$\mu^{(i)}_{k|k} = \mu^{(i)}_{k|k-1} + K^{(i)}_k\left(z - H_k\left(\mu^{(i)}_{k|k-1}\right)\right) \tag{12}$$

$$P^{(i)}_{k|k} = \left[I - K^{(i)}_k\nabla H_k\right]P^{(i)}_{k|k-1} \tag{13}$$

with ∇H_k being the Jacobian of the measurement equation with respect to the obstacles estimated location.

4.2 Algorithm

In order to use the presented framework, it is necessary to create Gaussians according to the measurements of the robot's sensors. Therefore $b_k(m|Z_{k-1}, X_{k-1})$ is obtained from the measurements Z_{k-1} and the previous robot position X_{k-1}. The components of this Gaussians are determined according to

$$\omega^{(j)}_{b,k} = 0.01, \qquad \mu^{(j)}_{b,k} = h^{-1}\left(z^j_{k-1}, X_{k-1}\right),$$
$$P^{(j)}_{b,k} = h'\left(\mu^{(j)}, X_{k-1}\right)R\left[h'\left(\mu^{(j)}, X_{k-1}\right)\right]^T$$

where h^{-1} is the inverse measurement equation, R is the measurement noise covariance and $h'\left(\mu^{(j)}, X_{k-1}\right)$ is the Jacobian of the measurement model function with respect to the Gaussian state, j. Therefore, the implementation initially considers all detections at time $k-1$ to be potential new features at time k.

Then, as every Gaussian is combined with every measurement to generate a new Gaussian, the numbers of Gaussians grow exponentially in every frame. That is why pruning and merging operations are necessary. Gaussians which are determined sufficiently close (through a Mahalanobis distance threshold) are merged into a single Gaussian. But this does not represent an elimination of an obstacle because one Gaussian can represent more than one obstacle by its weight; these values are added when two or more Gaussians are merged.

Figure 3 shows the pseudo code of the complete algorithm.

```
//prediction step
// the parameters of the Map's MoG model ($v_{k-1}(m|X_{k-1})$) are modified
for $i = 1$ to $J_{k-1}$ do
  //the obstacles may move -> covariance is increased
  $\mu_{k|k-1}^{(i)} = \mu_{k-1}^{(i)}, P_{k|k-1}^{(i)} = P_{k-1}^{(i)} + Q, \omega_{k|k-1}^{(i)} = \omega_{k-1}^{(i)}$
end for
//birth; new obstacles are added
generateNewGausians($Z_{k-1}, X_{k-1}$) // equation (5)
$v_{k|k-1}(m|X_k) = \left\{\mu_{k|k-1}^{(i)}, P_{k|k-1}^{(i)}, \omega_{k|k-1}^{(i)}\right\}_{i=1}^{J_{k|k-1}}$
//update step
for $i = 1$ to $J_{k|k-1}$ do
  calculate $P_D^{(i)}$
  $\omega_k^{(i)} = \left(1 - P_k^{(i)}\right)\omega_{k|k-1}^{(i)}$
end for
$N = 1$
for each $z$ in $Z_k$
  for $i = 1$ to $J_{k|k-1}$ do
    calculate $H, S_k^{(i)}$ and $K_k^{(i)}$
    $\mu_k^{(N+i)} = \mu_{k|k-1}^{(i)} + K_k^{(i)}\left(z - \mu_{k|k-1}^{(i)}\right)$
    $P_k^{(N+i)} = \left[I - K_k^{(i)} H\right]P_{k|k-1}^{(i)}$
    $\tau^{(i)} = P_D^{(i)}\omega_{k|k-1}^{(i)}\left|2\pi S_k^{(i)}\right|^{-0.5} \times \exp\left(\left(z - \mu_{k|k-1}^{(i)}\right)\left[S_k^{(i)}\right]^{-1}\left(z - \mu_{k|k-1}^{(i)}\right)^T\right)$
  end for
  for $i = 1$ to $J_{k|k-1}$ do
    $\omega_k^{(N+i)} = \tau^{(i)}/\left(c(z) + \sum_{l=1}^{J_{k|k-1}} \tau^{(l)}\right)$
  end for
  $N = N + J_{k|k-1}$
end for
$J_k = N$
//updated map
$v_k(m|X_k) = \left\{\mu_k^{(i)}, P_k^{(i)}, \omega_k^{(i)}\right\}_{i=1}^{J_k}$
prune ($v_k(m|X_k)$)
```

Fig. 3. Pseudo code of the general algorithm that calculates the PHD that represent the map of

obstacles.

With this algorithm, the PHD of the map is obtained. Then to get the position of the obstacles in the map, it is necessary to evaluate every Gaussian's weight. If this values exceeds a given threshold, then the obstacle position is given by the Gaussian's mean vector, and it's added to a vector that represent the current map. Figure 4 shows this algorithm.

```
$\mathrm{M}_k = [\,]$
for $i = 1$ to $J_k$ do
    if $\omega_k^{(i)} > thrld$ then
        $\mathrm{M}_k = [\mathrm{M}_k \ \mu_k^{(i)}]$
    end if
end for
```

Fig. 4. Pseudo code of the algorithm that drawn obstacles according to the PHD of the map.

4.3 Application

Using the proposed methodology, it is obtained a representation of the obstacle map for the detection of soccer players (Nao robots) in the SPL league. To do this, the methodology is used as follows:

i. State space: in order to describe the obstacles in the field, the state space is a vector $p = (x, y)$ that represents positions on the field according to the center of the field as $(0,0)$ of the coordinate system.

ii. Sensor: the used sensor is the Nao camera. This implies that transformations must be done in order to describe the measurements as positions on the field, using the camera's intrinsic and extrinsic parameters, as well as the position of the robot on the field. The detections are made with the same robot´s detector provided in the B-Human Code Release 2014 [14]

iii. Probability of detection P_D: Given that the sensor is the camera of the robot, the probability of detection is given by the field of view of it and the position of the obstacle relative to the robot. This implies that P_D must be recalculated in every frame for all the Gaussians of the map.

iv. Moving obstacles: The movement of the obstacles is taken into account by growing the covariance of the mixture of Gaussians in every frame instead of adding a movement model into the prediction step.

5 Results

In order to evaluate the proposed methodology several experiments with real Nao robots in a real SPL field were carried out. Given that we needed to measure the accuracy of the obstacle's map (i.e. robots map), a validation system consisting of a global vision system (camera over the field) for measuring the Ground Truth was implemented.

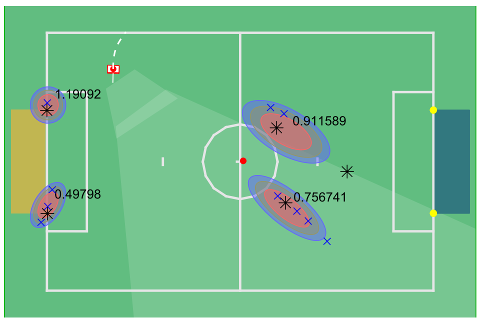
First, for very simple initial conditions, we carried out only one experiment in which a robot is placed in the center of the field and it observes three other static robots. The robot is moving its head all the time, and given its reduced field of view, at a given moment it is able to observe just one of the other robots and in some few cases two. The proposed GM-PHD based method is compared with a classical EKF based method. As expected, given the simplicity of the problem, both systems obtained an average error of about 20 cm in the position of the robots. Both methods run in real time, being the processing time of the GM-PHD method 0.13 ms, and the processing time of the EKF method 0.07 ms.

Secondly, the proposed GM-PHD based method and the classical EKF based method were compared in a set of experiments under a variety of more realistic and dynamic conditions, where the observer robot, i.e. the one that builds the map, moves as well as some of the observed robots. Figure 6 shows this set of experiments.

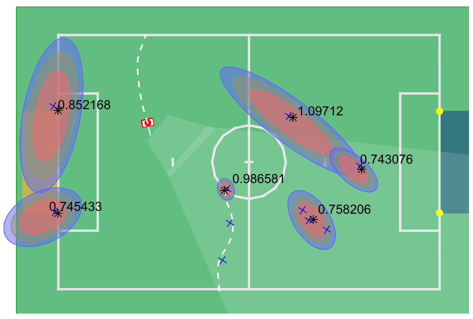
For the first experiment of this set, five static robots are placed on the field, and the observer robot performs a ready positioning, i.e. the robot walks from its starting position to their legal kick-off position. The observer robot moves its head from left to right all the time, hence the other robots are not inside the FoV in every frame. As can be seen in Fig. 5(a), the differences of the GM-PHD method and the classical EKF method, in term of a multi-tracking criteria, are notorious. While the GM-PHD approach correctly describes the presence of obstacles in most of the positions of the field, the classical one shows an incorrect number of obstacles for each real one. This is because the new observations are not correctly associated with the previous ones, due to the odometry errors and the non-constant observations; then new hypothesis are drawn incorrectly by the EKF method.

In the second experiment one moving robot observes five other robots; one moving robot and four static ones. The observer robot, while moves, observes the other moving robot occasionally, because it moves its head from left to right all the time. In Fig 5(b) it can be seen that, when using the classical EKF approach, there are two wrongly detected robots placed in the previous path of the moving robot, in addition to the same error that occurs in the last experiment when more than one obstacle in the map is describing each real one. The GM-PHD method correctly relates these observations with the same obstacle. In fact, the GM-PHD method perfectly estimates the number of robots in the field. In the case of the EKF method, bad associations can be corrected by increasing the minimal distance of merging. But this can produce another type of errors, where detections from different robots are associated to the same one.

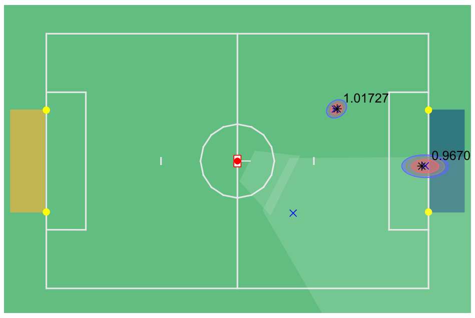
In the third experiment we analyze a typical kidnapping situation. The observer robot is placed in the center of the field and three static robot are placed in other field positions. The observer robot is looking around when one of the static robots is removed from the field (in a real match, this is very common due to robot penalizations). As can be seen in Fig. 5(c) and Fig. 5(d), the GM-PHD approach deletes very quickly the hypothesis associated with the kidnapped robot, while the classical EKF method keeps the track until the covariance reaches a given threshold value. This can be fixed for the classical EKF method by calculating a different rate



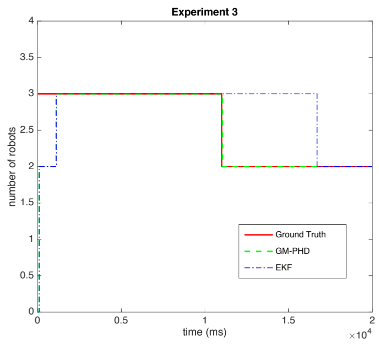
(a)



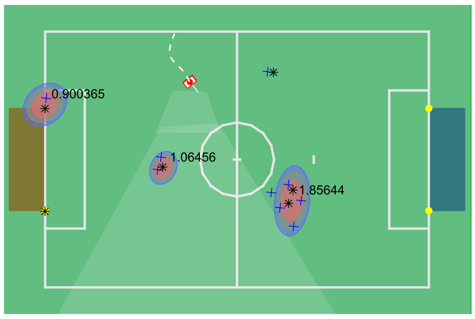
(b)



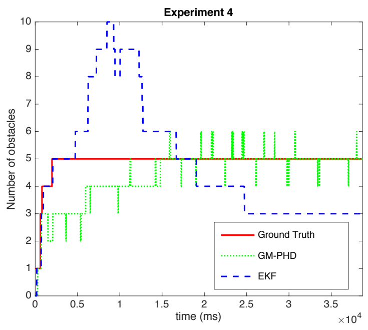
(c)



(d)



(e)



(f)

Fig. 5. Map building experiments under dynamic conditions. Four different situations are described in (a), (b), (c) and (e). In these diagrams the black asterisks represent the real position of the robots, obtained by the Ground Truth system; Blue crosses represent the robots' positions calculated by a EKF tracking method; The colored ellipses represent the robot estimations of the GM-PHD based method, and the associated number represents the weight of each Gaussian. The white dashed lines represent the trajectory of moving robots. (d)/(f) shows the estimated number of robots corresponding to situation (c)/(e).

of covariance growing when a hypothesis that should be seen is not seen. But, this implies including another heuristic to the process, while the GM-PHD method handles this situation naturally.

Finally, in the last experiment the observer robot also realizes a ready positioning while there are some static robots placed in the field. But two of these robots are very close from each other, therefore the perception of these robots is very inaccurate. In Fig. 5(e) it can be seen that even when only one Gaussian is representing these robots, the GM-PHD method can correctly estimate the number of robots in that place (given by the weight of the Gaussian), while the classical EKF approach fails due the odometry and perception errors. In Fig. 5(f) the estimated number of robots given by each method thought the entire experiment is shown. It should be remembered that the estimated number of robots is calculated as the sum of the weight of all Gaussians by the GM-PHD method, and as the number of obstacles created by the classical method.

6 Conclusions

This paper presents a new method for building obstacle maps using a consistent mathematically approach, known as Random Finite Sets. The method is applied to the problem of estimating the position of the robots, teammates and opponents, in the SPL league. Considering the computational capabilities of Nao robots, the GM-PHD implementation is used. In this implementation, obstacles and observations are represented using Mixture of Gaussians, but instead of associating an obstacle or an observation to a given Gaussian, the weight of each Gaussians maintains an estimation of the number of robots that it represents.

The proposed tracking method was validated in several real game situations, with moving robots, and compared with a classical EKF based approach. The proposed GM-PHD method showed a much better performance, being able to deal with most of the data association problems, even being able to manage complex situations such a robot kidnapping. Moreover, the method is able to run in real-time in the Nao robots (mean processing time is 0.13 ms; worse case processing time 0.3 ms).

Acknowledgments

The authors thank Constanza Villegas for her contributions to the development of this publication and the UChile Robotics Team for their general support. We also thank the B-Human SPL Team for sharing their code release, contributing the development of the Standard Platform League. This work was partially funded by FONDECYT Project 1161500.

References

1. Mendoza, J.P., Biswas, J., Cooksey, P., Wang, R., Klee, S., Zhu, D., Veloso, M.: Selectively Reactive Coordination for a Team of Robot Soccer Champions. In: Proceedings of AAAI-16 (2016).
2. Trevizan, F.W.F., Veloso, M.M.M.: Learning Opponent's Strategies In the RoboCup Small Size League. In: Proceedings of AAMAS 2010 Workshop on

Agents in Real-time and Dynamic Environments. pp. 45–52. , Toronto (2010).

3. Yasui, K., Kobayashi, K., Murakami, K., Naruse, T.: Analyzing and Learning an Opponent's Strategies in the RoboCup Small Size League. In: Behnke, S., Veloso, M., Visser, A., and Xiong, R. (eds.) RoboCup 2013: Robot World Cup XVII. pp. 159–170. Springer Berlin Heidelberg, Berlin, Heidelberg (2014).
4. Laue, T., Röfer, T.: Integrating Simple Unreliable Perceptions for Accurate Robot Modeling in the Four-Legged League. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., and Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. pp. 474–482. Springer Berlin Heidelberg, Berlin, Heidelberg (2007).
5. Fabisch, A., Laue, T., Röfer, T.: Robot Recognition and Modeling in the RoboCup Standard Platform League. In: Pagello, E., Zhou, C., Behnke, S., Menegatti, E., Röfer, T., and Stone, P. (eds.) Proceedings of the Fifth Workshop on Humanoid Soccer Robots in conjunction with the 2010 IEEE-RAS International Conference on Humanoid Robots. pp. 65–70. , Nashville, TN, USA (2010).
6. Goodman, I.R., Mahler, R.P.S., Nguyen, H.T.: Mathematics of Data Fusion. Springer Netherlands, Dordrecht (1997).
7. Mahler, R.P.S.: A Theoretical Foundation for the Stein-Winter "Probability Hypothesis Density (PHD)" Multitarget Tracking Approach. In: Sensor and Data Fusion (2000).
8. Mahler, R.: A brief survey of advances in random-set fusion. In: 2015 International Conference on Control, Automation and Information Sciences (ICCAIS). pp. 62–67. IEEE (2015).
9. Mahler, R.P.S.: Statistical Multisource-Multitarget Information Fusion. (2007).
10. Mahler, R.P.S.: Multitarget Bayes Filtering via First-Order Multitarget Moments. IEEE Trans. Aerosp. Electron. Syst. 39, 1152–1178 (2003).
11. Vo, B.N., Singh, S., Doucet, A.: Sequential Monte Carlo methods for multi-target filtering with random finite sets. In: IEEE Transactions on Aerospace and Electronic Systems. pp. 1224–1245 (2005).
12. Vo, B.-N., Ma, W.-K.: The Gaussian Mixture Probability Hypothesis Density Filter. IEEE Trans. Signal Process. 54, 4091–4104 (2006).
13. Random Finite Sets for Robot Mapping & SLAM - New | John Stephen Mullane | Springer, http://www.springer.com/gp/book/9783642213892.
14. Thomas, R., Laue, T., Judith, M., Bartsch, M., Batram, M.J., Arne, B., Martin, B., Kroker, M., Maaß, F., Thomas, M., Steinbeck, M., Stolpmann, A., Taddiken, S.: Team Report and Code Release 2013. 1–194 (2014).