2016 IEEE International Conference on Multisensor
Fusion and Integration for Intelligent Systems (MFI 2016)
Kongresshaus Baden-Baden, Germany, Sep. 19-21, 2016

Multi-Object Tracking with Distributed Sensing*

Ricardo Dias[1,2], Nuno Lau[1,3], João Silva[1,4] and Gi Hyun Lim[1,5]

Abstract— **One of the main research goals on distributed autonomous agents in a Multi-Agent System is the development of mechanisms to form a better world model using information merging from different agents. In this paper, we present a solution for robust online and real-time multiple object tracking in a multi-agent system using information gathered by various agents over time, using COP-KMeans for clustering and Kalman Filtering for object state estimation. The proposed solution was implemented on a real robotic soccer team and evaluated in the RoboCup Middle-Size League competitions. The robotic soccer was presented as one possible application for the ideas expressed on this paper.**

I. INTRODUCTION

One of the main research goals on distributed autonomous agents in a Multi-Agent System (MAS) [1] is the development of mechanisms to form a better world model using information merging from different agents. It has already been demonstrated that distributed sensor fusion can enhance the belief by synergistically merging data from different agents [2] to derive a better approximation of the World Model than would be possible with each one individually [3].

When compared to centralised approaches, distributed systems present a major advantage: they are usually more resilient to failures. However, it is much more difficult to implement a real fully distributed system, when comparing with a centralised approach, in which one “master“ agent takes control of a task that all the team will benefit from.

In this paper, we present a solution for robust real-time multiple object tracking in a multi-agent system using information gathered by various agents over time. For the purpose of this paper, we assume that each agent is able to detect object candidates and will focus on the integration of these observations into their world model.

The proposed solution was implemented on a real robotic soccer team and evaluated in the RoboCup Middle-Size League world championships. While having a huge potential for a variety of applications, MAS are extensively tested and benchmarked in RoboCup. The RoboCup Middle-Size League provides an excellent testbed for autonomous robotic teams in stochastic and highly dynamic environments. A soccer match cannot be overlooked as a testbed, since it resembles more the real world (complex and semi-unstructured) than a research lab. Although robotic soccer is presented as an example, the ideas expressed on this paper are not limited to this application area.

Following this Section, which introduces the problem and some concepts about the application, we will start by defining our implementation of the local object tracking in Section II that will define how tracking is done in each agent individually. In Section III, we will present a methodology used to merge information from multiple agents to form a unified representation of the obstacles spread around the field. Then, we show some results and respective discussion in Section IV and conclude the paper in Section V.

A. RoboCup Middle-Size League

Among the RoboCup leagues, the Middle-Size League (MSL) is one of the most challenging in terms of rules and environment (Figure 1). In this league, robots play soccer autonomously in a 18x12m field with a standard size-5 FIFA ball. Each team can have up to 5 robots with maximum size of 50x50 cm base and 80 cm height and are not allowed to weight more than 40 kg. Currently, all teams participating in the league have an omni-directional drive system. The rules of the matches are based on the official FIFA rules, with a few required changes to adapt for the playing robots.

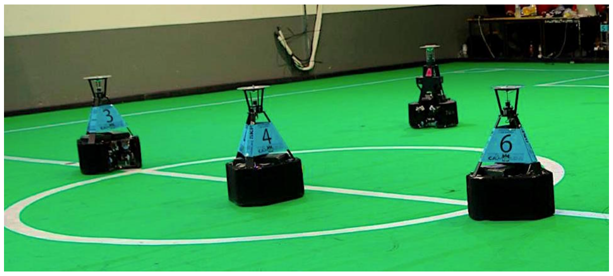

Fig. 1. Middle-Size League final in RoboCup Portuguese Robotics Open at Bragança, Portugal

Specifically in this league and regarding sensor fusion for world modelling [4], teams have been applying Kalman Filters for ball and obstacles state estimation [5], [6]. Moreover, some interesting work has been done with integration of information from several agents [7], in which consensus problems are applied to distributed sensor fusion [8] and belief propagation [9], for instance.

B. The CAMBADA Team

The developed work was accomplished within the MSL context, more specifically in the CAMBADA (Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture) team [10], the MSL Robotic Soccer team from

*This research project was funded by National Funds through the Foundation for Science and Technology, in the context of the project UID/CEC/00127/2013, and Euroc FP7-2013-NMP-ICT-FoF 608849

[1]DETI/IEETA, University of Aveiro, Aveiro, Portugal
[2]`ricardodias@ua.pt`
[3]`nunolau@ua.pt`
[4]`joao.m.silva@ua.pt`
[5]`lim@ua.pt`

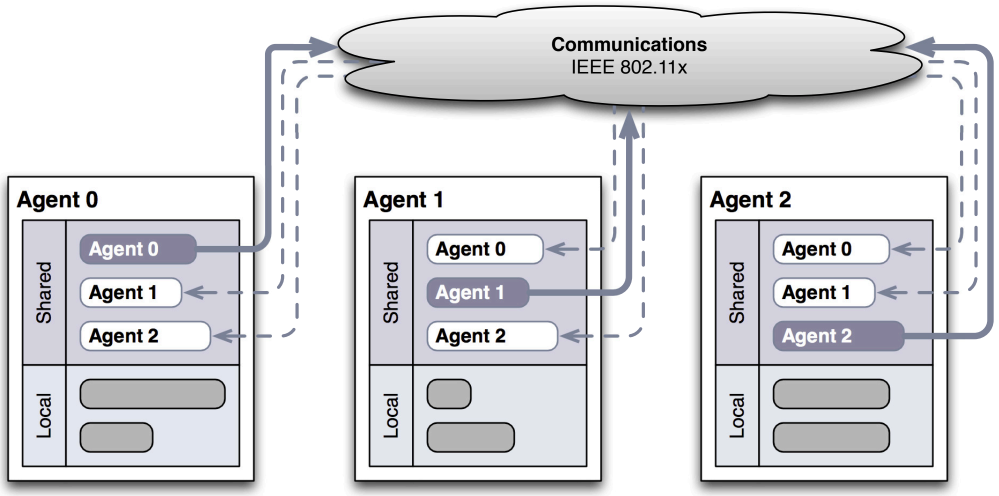

Fig. 2. Example of RtDB functionality with 3 agents. Rtdb items can be shared (broadcasted to all team members) or local (can be used in processes running in the local agent)

the University of Aveiro. This project started in 2003 and is currently coordinated by the IEETA IRIS group and involves people working on several areas from hardware (building the mechanical structure of the robot, its hardware architecture and controllers) to software (image analysis and processing, sensor and information fusion, reasoning and control).

In this Section we present two main components of the team software architecture that are relevant for the scope of this work: the communication model and the agent integrator.

1) Communication Model: The Realtime Data Base (RtDB) [11] is an open-source middleware developed by the CAMBADA team that provides a seamless access to the complete team state using a distributed database, partially replicated to all team members, as is depicted on Figure 2.

All the information that the team needs to share (such as the absolute positions and poses of all players, as well as the position of the ball, etc.) are included in the RtDB, which is extremely useful while in-game. For example, when a certain robot does not see the ball, it can use other teammate's information as a guide. The modular structure of the RtDB allows to easily add and remove items from it via a configuration file.

The RtDB can also be intensively used as an inter-process communication mean, due to its capability of defining local memory items that are not broadcasted to other robots, but are still accessible by other processes running on the same agent environment. This local information may include sensorial data from the vision system and the hardware platform and also commands that are sent to the low-level control layer through some gateway hardware interface.

The RtDB is tightly coupled with one process called "comm", which handles the Wi-Fi communication for both sending information to the multicast group and receiving data from other agents [12]. Together, RtDB and "comm" are used by CAMBADA (and several other teams in MSL) to solve the issues of intra-robot and inter-robot communications at the same time, in a flexible and modular way.

2) Agent Integrator: Values provided by any sensor are inevitably noisy and usually not very usable in their raw form. This is why the integration of the information available to the robot is a crucial step.

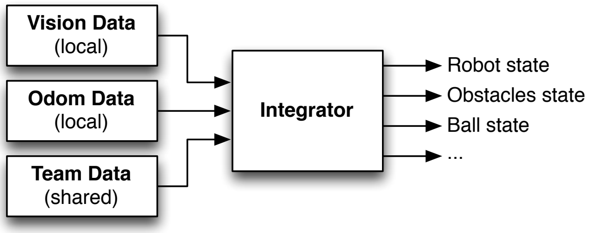

Fig. 3. Illustration of the agent integrator module

In the case of the CAMBADA software architecture, this is the function of the Integrator module, which takes values from different sources (camera, wheel encoders, teammates shared info, etc.), which are present in the RtDB, and properly handle them so that a relevant and trustworthy database is built to represent the current state of the world (Figure 3).

When this work started, the agent was able to successfully create a representation of the obstacles around the robot at each agent cycle, but no merging was done among the team members to take advantage of the distributed sensing and communication model in this environment. If a correct representation of the opponent obstacles can be achieved, then the team can evolve from simple obstacle avoidance to more complex areas such as opponent modelling and coercive behaviour.

One of the main problems is the fact that robots move very fast (up to 4 meters per second), which introduces a lot of noise in its measurements, thus inducing a lot of false detections. Therefore, a set of heuristics needs to be applied to validate these detections.

II. Single-Agent Multi-Object Tracking

In a first instance, the integrator has to work with the local information it gets in the current cycle, only afterwards it is able to integrate information from other agents. In this Section, we show how we implemented our Object Tracking software module and discuss its usage in both obstacle tracking and ball tracking.

A. Track Definition

In this work we assumed that each track can be modelled as a Gaussian process in the 2D space, so each will contain a Kalman Filter with 4 state variables: x, y, $\dot{x}$, $\dot{y}$ - position and velocity in x and y axis, of which implementation is beyond the scope of this paper. Since robots are omni-directional, orientation was not considered.

Furthermore, it is important to include some extra properties that will allow the tracker to monitor each track consistency and to purge old tracks:

- `age` - how many cycles ago has this track been created
- `visibleCount` - how many cycles this track has been visible since it was created
- `invisibleCount` - how many consecutive cycles this track has not been matched with an observation

B. Obstacle Tracking

Being soccer a very dynamic scenario, our robots need to be able to perceive the other robots around them, both opponents and their own team-mates. They are required to move around the field, either for re-positioning or dribbling the ball, while avoiding contact with any other obstacle on the field.

The simplest approach would be a reactive one, in which the actions of the robot are defined by the obstacles seen at that agent cycle. However, because the robots move and rotate at very high speeds, there is always a percentage of false-positives and false-negatives, which justifies the need for an obstacle tracking method.

Taking into account that robots in this league have omni-directional drive, there is no interest in estimating the obstacles orientation. On the other hand, in game situations, knowing the opponent positioning may allow the team to act quicker, anticipate their actions and even prevent dangerous situations like forward passes by covering an opponent from the ball perspective, so it is crucial that this information is as accurate as possible.

C. General Algorithm

For each new agent cycle, the object tracker algorithm is updated with the observations on that cycle. This algorithm can be summarized in 6 steps:

1) Predict new track positions
2) Assign observations to tracks
3) Update assigned tracks
4) Update unassigned tracks and purge old tracks
5) Create new tracks
6) Sorting

Step 1 *- Predict new track positions:* This step consists in running the predict step in each track's Kalman Filter. It will not only update the covariance matrices, but also the state estimate, predicting the position of each object one cycle ahead, using the filtered velocity.

Step 2 *- Assign observations to tracks:* For the purpose of this paper, we assume that each agent already has a set of obstacles observations at the present cycle.

This step presents a data association problem: some of these observations are obstacles which are already being tracked, others are newly detected obstacles that will originate the creation of new tracks and finally some tracks might not be updated with observations.

To solve this pairwise association problem, we used the Hungarian method [13], that finds an optimal assignment for a given cost matrix C in polynomial time, if we assume the constraint that only one observation exists per track.

The cost matrix that was used is a padded square matrix - it has additional columns/rows to account for the possibility of not matching any of the observations with the current tracks. If there are T tracks and N observations, the dimension of the cost matrix C will be $(T + N) \times (T + N)$. The lines of this matrix represent tracks and the columns represent observations.

After running the Hungarian algorithm over the cost matrix C, there will be three main groups in the posteriori logical matrix C': matched tracks (M), unmatched tracks (U) and unmatched detections (V). The lower-right part of the matrix C' is ignored.

$$C' = \left(\begin{array}{c:c} \mathrm{M} & \mathrm{U} \\ \hdashline \mathrm{V} & 0 \end{array}\right)$$

Step 3 *- Update assigned tracks:* For each match (submatrix M), the track's Kalman Filter is updated with the respective observation, `age` and `visibleCount` counters are incremented and `invisibleCount` is reset.

Step 4 *- Update unassigned tracks and purge old tracks:* For the unassigned tracks (submatrix U) The `age` and `invisibleCount` counters are incremented and a factor $f < 1.0$ is applied to the velocities, to avoid that tracks which will remain unassigned for some cycles (and eventually removed later) remain with the same speed during that period. These factors are adjusted based on the dynamics of the opponent team and our team frame-rate.

Furthermore, a validation is performed to check if the track should remain on the list or should be removed. Here, two heuristics were applied: a track is deleted if either the `invisibleCount` counter is higher than a threshold or the visibility ratio (`visibleCount/age`) goes below a certain threshold. The first heuristic is the simple purge method, in which a track is deleted when it is not observed for a while. The second one allows spurious detections to be deleted on the next cycle (even if the invisible threshold was still not reached).

Step 5 - *Create new tracks:* A new track is created for each unassigned detection (submatrix V), which represent observations not matched with any prior track.

Step 6 - *Sorting:* In the case of obstacles, the sorting is done by descending visibility - (tracks with higher visibility ratio will be first on the list). This step will be important later, for the decision of which tracks should be shared with the team-mates, since there is a limit on the number of tracks that can be shared, imposed by MSL rules as a bandwidth usage limit.

D. Obstacle Sharing Criteria

Despite locally detecting and taking into account every observation in its decisions, each agent is very cautious about the information it shares. Since other agents on the team can use the shared information, it is very important that the information that an agent broadcasts is as accurate as possible and ideally with no false-positives.

Therefore, a set of conditions are required to start sharing a particular track on the RtDB. As previously described, tracks with higher visibility have priority to be shared. Furthermore, it is required that the track `age` is higher than a threshold, it must have a minimum `visibleCount` and visibility ratio, and must not exceed a maximum distance from the observing robot (to prevent false positive detections on higher distances).

Once a track is set to be shared, it will be shared until it is deleted or leaves the field.

E. Ball Tracking

The ball is one of the most important objects on the field, since the objective is to play soccer. Just like in the case of obstacles, it is also common to have spurious ball detections for various reasons, including occlusions by other robots, lighting conditions, high speeds (causing motion blur on the image), self occlusion while dribbling, etc.

Ball tracking differs from obstacle tracking in the fact that there should be just one ball inside the field, but the agent should be able to handle situations where it perceives more than one candidate. We decided to keep track of multiple balls, using the same method described above for obstacle detection. However, instead of sorting by visibility we select the oldest track as the one to use - so, in this case, the sorting is done by age instead of visibility. Different purging thresholds were also used, but the core of the method is the same as described in the last Section.

III. Multi-Obstacle Tracking With Multiple Observations

In this Section, we present a methodology used to merge information from multiple agents to form a unified representation of the obstacles spread around the field.

In the context of the Middle-Size League, this is particularly important for the coach, which is a computer allowed to communicate with the robots, but not allowed to have any sensors attached. The objective of this coach is to give high-level coordination instructions for the team - strategy, formation, attitude, etc. Using the information shared by the robots, the coach is able to create a representation of the opponents position, which can be used to anticipate opponent gameplay, such as forward passes and set-plays.

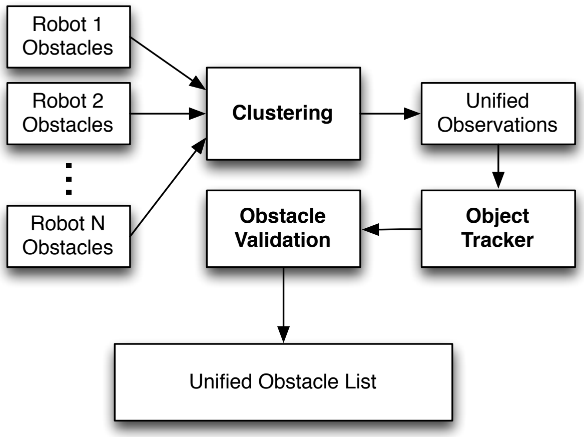

Fig. 4. Summary of the tracking system using multiple agent shared observations

Figure 4 shows an overview of the global tracker. It considers the various agents shared obstacle tracks as observations. Although this shared information is not made of raw observations, but rather processed observations that have been associated together as a track and met the previously discussed criteria to be shared among the team, they can be considered observations for the purpose of creating this unified obstacle list.

A. Observation Clustering

In the last Section, we already demonstrated how the Hungarian Algorithm can be used to match observations with tracks. However, by solving a global minimization problem, it can not account for situations where there are multiple observations for the same object. Therefore, a clustering algorithm was implemented, based on the Constrained K-Means method [14], to take advantage of the background knowledge, that can be expressed as a set of instance-level constraints on the clustering process.

In the case of the MSL, the maximum size of the obstacles is limited by the rules (50×50cm), which implies that an obstacle that occupies the maximum allowed size can be perceived as a 70.7cm-wide obstacle (when seen from the diagonal). Despite this theoretical value, on this league, at the time of writing this paper, most teams opt for a triangular configuration on their platforms, meaning that size is not reached. Moreover, their sides do not measure less than 30cm.

Using this background knowledge, the *COP-Kmeans* method has been applied with the following constraints:

- if $width(centroid_i) > 0.7$m, split in two centroids
- if $distance(centroid_i, centroid_j) < 0.3$m, merge $centroid_i$ with $centroid_j$

B. Applying the Object Tracker

The output of the previous clustering stage is a unified observation list, which is the input for the object tracker module. Its implementation was already described in Section II-C.

C. Obstacle Validation

To avoid unreliable information, the unified tracks are subject to a validation. Once a track has been validated it will remain valid until it disappears.

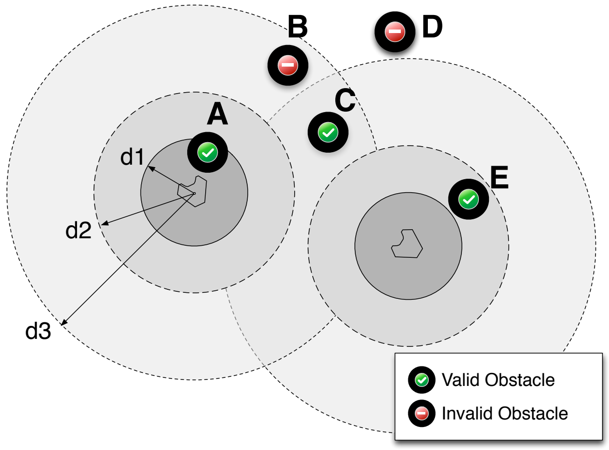

Fig. 5. Example of the validation criteria using two robots. Here, three observations are validated and two do not pass the validation criteria. The image is not in scale.

Figure 5 shows an illustration of the implemented validation methodology. Essentially, the position of the team robots define three different zones:

- **Zone 1** - any position closer than d_1 from a team mate. In this zone, only tracks observed by the closest team mate robot can be validated. It is such a small distance that the closest robot must be able to see it directly. This prevents using detections of spurious obstacles close to team-mate positions.
- **Zone 2** - any position closer than d_2 from a team mate. Any track lying on one of these zones is validated.
- **Zone 3** - any position closer than d_3 from a team mate. A track lying on this zone requires at least two observing robots to be validated.
- Any tracked obstacle which distance from the closest team-mate is more than d_3 is not validated, since it is outside the maximum detection distance boundary.

In the same figure, there are 5 obstacles:

- **Obstacle A - Valid** - its position lies inside **Zone 1** of the robot on the left.
- **Obstacle B - Not Valid** - the obstacle is inside **Zone 3** of one robot, but not the other robot.
- **Obstacle C - Valid** - it is inside **Zone 3** of both robots.
- **Obstacle D - Not Valid** - It is outside all zones of all robots.
- **Obstacle E - Valid** - it is inside **Zone 2** of a team-mate.

IV. Results and Discussion

In MSL, since 2016, teams are encouraged to supply their worldstate information during the matches (only for logging purposes), with the objective of providing the other team a reference to benchmark solutions after the game. In this case, knowing the exact location of the opponents after the match, allows us to use it as a groundtruth to match our obstacle detection algorithm.

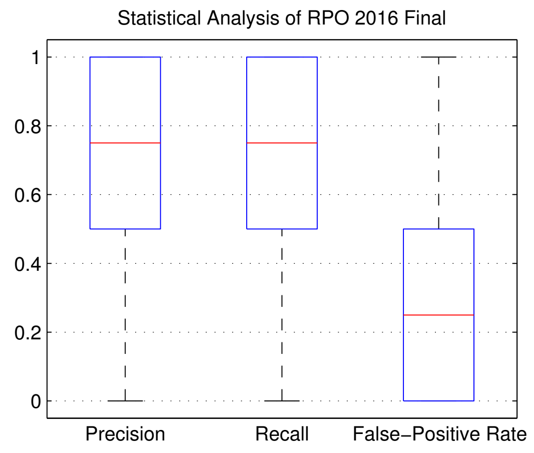

Fig. 6. Detection Rate and False-Positive Rate results

Therefore, we implemented and tested this approach during the first half of the final of the RoboCup Portuguese Open 2016 and analysed our obstacle detection against the "groundtruth" provided by the other team. For this purpose, we only considered free-play situations, because in other situations, the referee may be inside the field repositioning the ball, and therefore inducing extra obstacles in the perception of the robots. Under these conditions, our dataset contains **1828 frames** sampled at **10 Hz**. Based on the false-positive rates and considering the predefined rules for validation on zones 1,2 and 3, the considered distance parameters were $d_1 = 1.0$m, $d_2 = 2.5$m and $d_3 = 5$m.

To benchmark the performance of this solution, we used two different metrics:

- **Precision** - percentage of correctly identified obstacles (over the detections) in each frame
- **Recall** - percentage of correctly identified obstacles (over the groundtruth) in each frame
- **False-Positive Rate** (FPR) - percentage of outliers in each frame

As Figure 6 shows, we obtained a median of:

- **Precision**: 75%
- **Recall**: 75%
- **False-Positive Rate**: 25%

Of course, the objective is always to maximise both precision and recall and to minimise the FPR. However, there is always a trade-off between between these two metrics, since it is always possible to increase distances d_1, d_2 and d_3, but not without sacrificing the FPR, because increasing those distances would mean to detect even more false-positives.

Experience and visual analysis tell us that these spurious detections occur mostly when our robots are moving fast (they can reach velocities of 4 meters per second) and the obstacles are far away from the detecting robots. As an improvement, the distances could vary with the robot velocity modulus.

It is also important to note that we are including false-positives that are created at the beginning of all this process - object detection - of which scope is outside this work. Moreover, by visual inspection we also noticed that sometimes the false-positive detections occur near our robots. Given our premises, it means that the robots sometimes wrongly identify obstacles in its vicinity. This can occur, for example, in cluttering situations, where strong shadows appear on the field. Since our obstacle detection relies mostly on colour (and the robots have to be mostly black), this sometimes creates false detections.

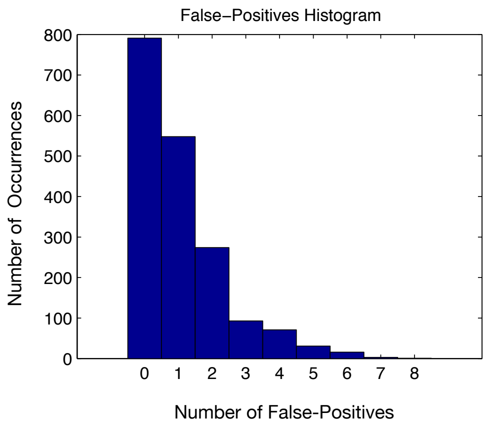

Fig. 7. Histogram of the number of False-Positives

However, we further investigated the false-positives and the histogram in Figure 7 shows that we detected at most 8, but that most times the number stays between 0 and 2, which is acceptable, given the environment conditions of the test. In fact, in our previous approach (a naive merging algorithm, without validation), the robots could detect up to 20 obstacles in some situations, which would give us, at least, 15 false-positives. Unfortunately, there is no work from other teams which we can fairly compare these results with.

The evaluated metrics will allow us to further improve the algorithm, by optimizing the avaliable parameters, always with the objective of maximising the detection rate and minimising the False-Positive Rate.

V. Conclusion

In this paper we presented a solution to integrate information from several agents to formulate a unified world state representation. We started by discussing the implementation of an object tracking module and its application in tracking two different types of objects in a robotic soccer environment: obstacles and the ball.

We then moved to the presentation of a methodology to merge this information that is generated in (and shared by) different agents into an unified representation of the obstacles spread around the field.

This solution was implemented on a Middle-Size League agent integrator and tested during the RoboCup Portuguese Open 2016 competition.

The results show that a relatively high detection rate can be achieved, with some room for improvement concerning the false-positive rate. Nonetheless, this strategy played a major role in a number of situations, by allowing the team to act quicker, anticipate the opponent actions and even prevent dangerous situations like forward passes by covering an opponent from the ball.

References

[1] M. Wooldridge, *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[2] L. Merino, F. Caballero, J.-d. Dios, and A. Ollero, “Cooperative Fire Detection using Unmanned Aerial Vehicles,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, April 2005, pp. 1884–1889.

[3] W. Elmenreich, “Sensor Fusion in Time-Triggered Systems,” Ph.D. dissertation, Institut fur Technische Informatik, Vienna, Austria, 2002.

[4] J. Silva, N. Lau, A. J. Neves, J. Rodrigues, and J. L. Azevedo, “World modeling on an MSL robotic soccer team,” *Mechatronics*, vol. 21, no. 2, pp. 411–422, 2011, Special Issue on Advances in intelligent robot design for the Robocup Middle Size League.

[5] M. Lauer, S. Lange, and M. Riedmiller, “Modeling Moving Objects in a Dynamically Changing Robot Application,” in *KI 2005: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, U. Furbach, Ed. Springer Berlin Heidelberg, 2005, vol. 3698, pp. 291–303.

[6] X. Y., J. C., and T. Y., “SEU-3D 2006 soccer simulation team description,” in *CD Proc. of RoboCup Symposium 2006*, Bremen, Germany, 2006.

[7] R. Olfati-Saber, J. Fax, and R. Murray, “Consensus and Cooperation in Networked Multi-Agent Systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, January 2007.

[8] R. Olfati-Saber and J. S. Shamma, “Consensus filters for sensor networks and distributed sensor fusion,” in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC’05. 44th IEEE Conference on*. IEEE, 2005, pp. 6698–6703.

[9] A. W. Stroupe, M. C. Martin, and T. Balch, “Distributed sensor fusion for object position estimation by multi-robot systems,” in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 2. IEEE, 2001, pp. 1092–1098.

[10] A. Neves, J. Azevedo, N. L. B. Cunha, J. Silva, F. Santos, G. Corrente, D. A. Martins, N. Figueiredo, A. Pereira, L. Almeida, L. S. Lopes, and P. Pedreiras, *CAMBADA soccer team: from robot architecture to multiagent coordination*. Vienna, Austria: I-Tech Education and Publishing, January 2010, ch. 2, pp. 19–45.

[11] L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, and L. S. Lopes, “Coordinating distributed autonomous agents with a real-time database: The CAMBADA project,” in *Proc. of the 19th International Symposium on Computer and Information Sciences, ISCIS 2004*, ser. Lecture Notes in Computer Science, vol. 3280. Springer, 2004, pp. 878–886.

[12] F. Santos, L. Almeida, L. S. Lopes, J. L. Azevedo, and M. B. Cunha, “Communicating among robots in the robocup middle-size league,” in *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes in Artificial Intelligence. Springer, 2009.

[13] H. W. Kuhn and B. Yaw, “The hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, pp. 83–97, 1955.

[14] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, “Constrained k-means clustering with background knowledge,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 577–584.