Two level world modeling for cooperating using a multiple hypotheses filter

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Download date: 16. Nov. 2023

Two Level World Modeling for Cooperating Robots Using a Multiple Hypotheses Filter

J. Elfring, M.J.G. van de Molengraft, R.J.M. Janssen and M. Steinbuch

Abstract— **Robots increasingly operate in dynamic environments and in order to operate safely, reliable world models are indispensable. A world model is the robot's view of the world and contains information about obstacle locations and velocities. A two level algorithm is proposed. It is of particular use for teams of cooperating robots and the algorithm is based on a multiple hypotheses filter. Each robot features a low level world model with a fast update rate which can be used for obstacle avoidance. The local world models are combined to one global view of the world that is shared between all robots and can be used for the implementation of team strategies. Labeling and tracking is added to the multiple hypotheses filter in order to reduce the sensitivity to track loss in case of temporary occlusions of objects or false measurements. The algorithm was extensively tested during the 2010 RoboCup Middle Size League world championships in Singapore, the results of which are presented.**

I. INTRODUCTION

Autonomous systems are increasingly operating in dynamic environments. In order to operate safely, reliable navigation skills are crucial. For this purpose, an autonomous system needs knowledge about the locations of the objects around it. This type of knowledge is typically stored in a world model, which contains the robot's view of the world. In addition, a world model can contain room temperature, humidity, etc. In this paper, the focus is on world models that contain the location of a robot in its environment and the location and velocity of an unknown varying number of moving objects around it.

Building up a world model must be done in real-time on the basis of measurements that are performed either by sensors on the robot, *e.g.*, an onboard laser range finder, or sensors around the robot, *e.g.*, a camera on the ceiling of the room. The measurements can originate from multiple, possibly different sensors and from multiple robots. The number of objects is typically unknown and varying, since humans, robots or other moving objects, might enter and leave the robot's environment at any time. Determining whether measurements indeed represent a specific object or measurement noise (clutter) is called the data association problem. The problem of building a world model consisting of object locations and velocities is often referred to as multiple target tracking and localization (MTTL), see [1].

In literature, many MTTL algorithms solving the world modeling problem that is considered in this paper are intro-duced. The most promising algorithms are described below. For an extensive overview of MTTL algorithms the reader is referred to [1].

The first method is the Multiple Hypotheses Filter (MHF), introduced in [2] and used in, *e.g.*, [3]. With this filter, a tree that maintains all possible solutions to the data association problem is built and the corresponding probabilities are calculated. The hypothesis with the highest probability is selected as being the correct solution. An advantage is that multiple hypotheses are maintained, which enables revising the solution on the basis of new sensorial information. Furthermore, the MHF is "generally accepted as the preferred method for solving the data association problem in modern multiple target tracking (MTT) systems", according to [4]. However, the number of hypotheses in an MHF increases rapidly and therefore a pruning strategy has to be applied. Pruning should happen carefully, since elimination of the most probable hypotheses must be prevented. Even with pruning strategy, the algorithm has high computational cost and storage which is one of its main disadvantages.

A second method that is widely applied in literature is an advanced joint probabilistic data association filter (JPDAF), *e.g.*, Monte Carlo JPDAF introduced in [5] or sample JPDAF introduced in [6]. Within the Bayesian JPDAF, as explained in [7], the target location is calculated using a weighted average of validated measurements. Validation of measurements is based on prediction and the weights are probabilities that represent the probability that the measurement indeed is target originated. Contrary to the JPDAF, the sample JPDAF is able to deal with an unknown, varying number of objects and allows nonlinear process and measurement models, which clearly is beneficial. The sample JPDAF has low computational costs compared to the MHF but the JPDAF class of algorithms makes irreversible decisions. Once the data association algorithm fails to select the proper solution, correcting is hard. Furthermore, the algorithm does not deal well with different tracks that approach each other.

A third method is the Probability Hypothesis Density (PHD) filter, see [8], or the improved cardinalized PHD filter, see [9]. In the cardinalized PHD filter, a probability distribution is used for the number of objects and the sequential Monte Carlo implementation introduced in [10] allows for dropping linearity assumptions. An advantage is that this method, contrary to the MHF and the sample JPDAF, is able to deal with measurements that are originated from multiple targets. However, in order to be of practical use, assumptions or approximations are required in order to find a closed form solution. Furthermore due to the lack of ordering

in the finite sets that are used, the PHD based algorithms are not able to maintain a record of target identities.

The world modeling algorithm that is proposed in this paper is based on a Bayesian MHF and uses [11] as a source of inspiration. In [11], static objects are considered. Here, filtering is added in order to allow tracking dynamic objects. Furthermore the pruning mechanism is replaced and the particle filtering used for clustering is redundant due to the two level approach. More specifically, the contributions of this paper are:

1) A two level approach: Each agent builds a local low level world model on the basis of its own measurements. This low level world model runs at a high update rate (order 30 Hz) to facilitate obstacle avoidance. In addition, a global high level world model collects the information from local world models at different agents in order to generate one global view of the world. This high level world model can run at lower update rates and enables cooperation between the robots.

Both the local and the global world model are based on the same Bayesian MHF approach. The local world model effectively performs a huge data reduction on the total number of measurements, such that the global world model can run in real-time. This way, the CPU effort scales about linearly with the number of measurements. An additional advantage is that this global world model provides a robot with information about objects that are outside its own measurement range. Fig. 1 visualizes the two level structure.

2) A heuristic labeling strategy: Often, objects are invisible due to temporary occlusion, which makes it difficult to keep track of the individual objects. During target occlusion, the labeled object position is propagated and once the object is visible again, the label is recognized and its position and velocity are updated.

3) Real-time demonstrator with a team of robots: The two level world model is tested in the RoboCup Middle Size League (MSL). In the MSL two teams of five autonomous robots play soccer against each other. Each robot has an omnivision camera that allows the robot to search a part of the field for obstacles. However, with this camera the robot can not distinguish between opponents and peer players since they both appear as black blobs in the camera image. The field measures 12×18 [m] and the field lines are used to determine the robot's absolute position. The RoboCup MSL is an ideal testing environment for MTTL based world modeling algorithms since multiple cooperative robots are involved. They all have individual tasks, *e.g.*, avoid opponents, but also have to collaborate in order to win the match. The high velocities of the opponents (up to 4 [m/s]) make the environment very challenging. The two level approach as presented in this paper does not rely on any RoboCup related assumption.

This paper is organized as follows. In Section II, some more details about the two levels within the algorithm are given. Section III explains the algorithm itself and Section IV presents experimental results taken from the RoboCup MSL. The paper ends with conclusions and an outlook to
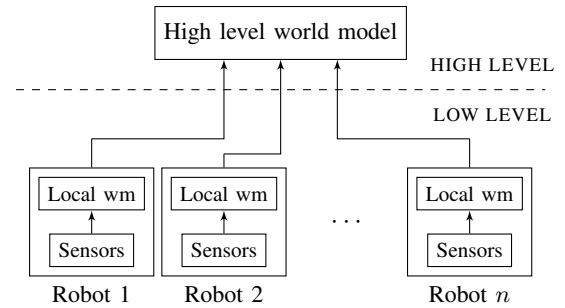

Fig. 1.   Schematic representation of the two level world model.

future work in Section V.

II. HIGH LEVEL VERSUS LOW LEVEL WORLD MODEL

The approach presented here explicitly takes into account that measurements can be performed by multiple agents and as a result, a two level strategy is proposed.

The low level part consists of a local world model that runs on each of the robots. It is assumed that each of the robots has at least one sensor that is able to measure 2D object locations $(z_x(k), z_y(k))$ relative to its own position, where k represents the time step of the measurement. These measurements are fed into the MTTL algorithm and the output of the algorithm is a local world model consisting of a collection of vectors $O^i(k)$:

$$O^i(k) = \left\{ \begin{pmatrix} x_1^i(k) \\ \dot{x}_1^i(k) \\ y_1^i(k) \\ \dot{y}_1^i(k) \end{pmatrix}, \ldots, \begin{pmatrix} x_{n_{\mathrm{obj}}}^i(k) \\ \dot{x}_{n_{\mathrm{obj}}}^i(k) \\ y_{n_{\mathrm{obj}}}^i(k) \\ \dot{y}_{n_{\mathrm{obj}}}^i(k) \end{pmatrix} \right\}, \quad (1)$$

where each state vector contains the absolute position and velocity of an object, n_{obj} represents the number of objects according to the algorithm, and i represents the agent number. In the remainder of the paper, the argument k is replaced by a subscript for ease of writing. This local world model typically runs at high update rates and is used for, *e.g.*, obstacle avoidance. Furthermore, it effectively performs a huge data reduction on the total set of measurements such that the global world model can run in real-time.

The high level part consists of a global world model that takes all object vectors generated by the local world models as input. A new collection of vectors, O_k^g, with object positions and velocities is the output. The global world model will mainly be used for strategy and, therefore, typically runs at a lower update rate. Furthermore, the global world model provides agents with information about objects that lie outside their own line of sight, but within the measurement range of fellow robots. In the RoboCup domain, the global world model is used for, *e.g.*, passing or global path planning towards the opponent's goal. Each individual robot runs its own version of the global world model. No centralized computer is available and running the global world model on one robot is undesired since the robots operate in an

aggressive environment. Breakdown of one robot would degrade the performance of all others. Ideally, all robots get the same input data from the low level world models which results in a consistent global world model.

III. ALGORITHM

This section gives a detailed explanation of the low level MHF algorithm. Sections III-A – III-E give a step by step explanation, whereas Algorithm III.1 gives a summary of the steps in pseudo code. The high level algorithm is identical, unless stated differently.

Algorithm III.1 Local world model algorithm

Input: New measurement $(z_x(k), z_y(k))$
Output: Collection of vectors $O^i(k)$

```
// Step A: Expand tree
for all hypotheses at time (k − 1) do
    new hypothesis = old hypothesis + new object;
    new hypothesis = old hypothesis + clutter;
    for all objects in current hypothesis do
        new hypothesis = old hypothesis + existing object;
    end for
end for

// Step B: Propagate and update if possible
for all hypotheses at time k do
    for all objects in current hypothesis do
        Propagate using constant velocity model;
    end for
    if (z_x(k), z_y(k)) associates with object then
        Update using constant Kalman gain observer;
    end if
end for

// Step C: Update probabilities
for all hypotheses at time k do
    Calculate p(z_k|h_{j,k}), p_c, p_n, p_e, using (3) − (4)
    Update probability using Bayes' law: (2);
end for

// Step D: Pruning hypothesis tree
for all hypotheses at time k do
    if C1 or C2 or C3 then
        Delete current hypothesis or object in hypothesis;
    end if
end for

// Step E: Select the best hypothesis
for all hypotheses at time k do
    Check p_{h_j(k)} and select h_best
end for
```

A. Expanding the hypothesis tree

Each time a new measurement arrives, the hypotheses tree is expanded. In the low level world model, this is a measurement performed by one of the robot's sensors, in the high level world model, such a measurement is the output of one of the local world models. In each of the levels, a measurement can either be (i) clutter, (ii) a newly appeared object, or (iii) an existing object. Each of these hypotheses has its own probability, p_c, p_n, and p_e respectively. This way, the first measurement generates two hypotheses:

h_1: The measurement represents a new object, or
h_2: The measurement results from clutter

and a second measurement generates five hypotheses:

h_1: Both measurements represent a new object
h_2: The first measurement represents a new object, the second measurement comes from the same object
h_3: The first measurement represents a new object, the second measurement results from clutter
h_4: The first measurement results from clutter, the second measurement represents a new object
h_5: Both measurements are clutter

Each time a new object is generated, a stationary Kalman filter with a constant velocity system model is initialized and attached to the object. The filter is initialized at the measurement location and initially has a zero velocity. This filter is an additional step compared to, *e.g.*, [11], and allows predicting and updating locations of moving objects.

B. Propagating and labeling of the objects

After expanding the tree, the filters are used to make a prediction of all object locations at the current time instant. The prediction is done using the predicted or updated object location and velocity after the previous measurement together with the constant velocity model.

For all objects associated with the latest measurement, the full state is updated. The update is done using a constant gain observer, where the gain is chosen to be the stationary Kalman gain under steady-state operation. A better choice for the filter gain would be to solve the Riccati equation online for each filter, but the high number of filters and the complexity of the algorithm complicate this alternative.

In the high level world model, this step also incorporates the heuristic labeling strategy used to distinguish between robots. Each robot knows its own position and its own unique identifier (ID). The measurement of its own location is very reliable and is labeled with a value equal to the robot's ID, measured locations of other robots do not have a label. Now the heuristic labeling update step is performed as follows:

- If the new measurement has a label ID and is associated with an existing object, the existing label is replaced by the robot's ID
- If the new measurement has no label and is associated with an existing object, the existing label is maintained
- If the new measurement has a label ID and is associated with a new object, the new object label has value ID
- If the new measurement has no label and is associated with a new object, the new object gets a new label
- If the new measurement is associated with clutter, no labeling is required

As a result of the first and third option two different objects within the same hypothesis can both have a label with the value ID. In that case the oldest object gets a new unique valued label while the newer label maintains its value ID. This effectively means that the newer measurement is considered to be more reliable. This heuristic labeling strategy has proven to be robust against temporary occlusions and enables the ability to distinguish between various opponents and peer players.

After this labeling, the distance between the measurement and the location of the object it is associated with is calculated. If the distance is too large, the association is considered to be unreliable and, therefore, the object is eliminated while the remainder of the hypothesis is maintained.

C. Updating and normalizing probabilities

In this step, the probabilities are updated using Bayes' rule:

$$p(h_{j,k}|z_k) = \frac{p(z_k|h_{j,k})p(h_{j,k}|h_{j,k-1})p(h_{j,k-1}|z_{k-1})}{p(z_k|z_{k-1})}, \quad (2)$$

where $p(h_{j,k}|z_k)$ is the probability of the current hypothesis given all measurements, $p(z_k|h_{j,k})$ is the probability of the latest measurement given the current hypothesis, $p(h_{j,k}|h_{j,k-1})$ is the probability of the current hypothesis given the previous hypothesis, $p(h_{j,k-1}|z_{k-1})$ is the probability of the previous hypothesis based on all measurements up to time $k-1$, the normalizing factor $p(z_k|z_{k-1})$ is the probability of the current measurement based on all previous measurement, and j is the hypothesis index.

Only hypotheses that associate the most recent measurement with an object include $p(z_k|h_{j,k})$ in (2) and this step is an alternative of the observation likelihood step in [11]. The distance between the position of the m-th object according to the filter state and the measurement, *i.e.*, $(x_m(k) - z_x(k), y_m(k) - z_y(k))$, is translated to a scaled probability:

$$p(z_k|h_{j,k}) = e^{-\frac{1}{2}\left(\frac{(x_m - z_x)^2}{\sigma_x^2} + \frac{(y_m - z_y)^2}{\sigma_y^2}\right)}, \quad (3)$$

where σ_x and σ_y are (tunable) standard deviations. Here $\sigma_x = \sigma_y = 0.3$ is chosen based on the expected radius of robots in combination with knowledge about the measurement inaccuracy. If the object detection algorithm is improved (increase measurement accuracy), or better system models and observer gains are used (improved prediction or update), these standard deviations can be decreased.

The probability $p(h_{j,k}|h_{i,k-1})$ depends on p_n, p_c, or p_e. These probabilities depend on the application and, as a first simple model, are chosen to be:

$$p_c = 10^{-3} \quad (4)$$
$$p_n = \alpha p_e \quad (5)$$
$$p_e = \frac{1 - p_c - p_n}{n_{\mathrm{obj}}}, \quad (6)$$

where α is a tunable parameter and n_{obj} is the number of objects in the hypothesis that is considered. The value α

is used to balance the probabilities p_n and p_e and depends highly on the environment. Within the RoboCup domain, objects come and go regularly and, therefore $\alpha = 0.1$ based on experience. Experimentally obtained data could be used to further improve these simple models.

The probability $p(h_{j,k-1}|z_{k-1})$ is known from the previous time step and the normalizing factor $p(z_k|z_{k-1})$ ensures that the probabilities of all actual hypotheses sum up to one.

D. Pruning of the hypothesis tree

The number of hypotheses grows more than exponentially in the number of measurements. Clearly, pruning of the hypotheses tree is inevitable to keep the algorithm maintainable. It is important that pruning happens carefully, since eliminating hypotheses is irreversible. In this algorithm, three criteria are checked:

C1: Is the probability of a hypothesis lower than a predefined threshold?
C2: Does the measurement fall outside a circular region with predefined radius relative to the sensor?
C3: Does the time since the last filter update exceed a predefined maximum time?

If C1 is answered with yes, the hypothesis is eliminated, if C2 or C3 is answered with yes the corresponding object is eliminated in all hypotheses while the hypotheses themselves are maintained.

In the first criterion it is assumed that once the corresponding hypothesis drops below a certain threshold, it will be likely not to be close to the real world situation. Here this threshold is chosen rather conservative as 1% of the highest probability occurring in the hypothesis tree. The second criterion is optional in the sense that it depends on the sensors that are used. In the RoboCup domain, the omnivision camera measurements are only reliable if the distance to the object is limited. If an object is not seen by the sensors for a pre-specified time, it is assumed that the object does not exist (anymore), as stated in C3. This maximum time should depend on the dynamics of the environment, *i.e.*, a highly dynamical environment such as RoboCup requires a low time.

In the high level world model, the second criterion is replaced by a criterion that ensures that at most a predefined number of hypotheses is maintained. If the number of hypotheses after the above mentioned pruning exceeds 1000, the least probable hypotheses are eliminated in addition. A less conservative pruning strategy would decrease the computational costs at the expensive of a higher risk of pruning valuable hypotheses.

E. Selection of the best hypothesis

In the last step, the hypothesis with the maximum a posteriori probability is selected. The corresponding object locations and velocities are the outputs of the algorithm.

IV. EXPERIMENTS

The two level algorithm that was explained in the previous sections has been extensively tested during the RoboCup

MSL world championship 2010 in Singapore. Team TechUnited became vice world champion and played all their ten matches with this world model, which means about five hours of test data. This section presents some illustrative results taken from the final, where the TechUnited soccer robots (called turtles) played against the Chinese team Water. A photograph taken during this match is shown in Fig. 2.

Fig. 2.   Photo taken during the final: TechUnited vs. team Water.

During the games, recordings were made using a perspective camera. A qualitative inspection of the output of the global world model, *i.e.*, the numbers of turtles and opponents and their trajectories, using this camera showed good correspondence.

Now, consider Fig. 3. The black dots represent the combined output of the local world models that run on the goal keeper and the defenders, *i.e.*, turtles 1, 3, and 4. Due to the limited surveillance range each turtle has, the field looks rather empty. Fig. 4 shows the output of the high level global world model that in addition to the data in Fig. 3 uses the data from the attackers. This gives a much more complete view of the field and illustrates why sharing data is highly advantageous. With the global world model, the defenders are able to position better and anticipate faster without having to explore the whole field on their own.

Next consider Fig. 5. This visualizes the output of the global world model, at a certain time t^*, combined with the black dots that are generated by the local world models. Based on the black dots shown in this figure, the algorithm introduced here generates results that might be debatable, especially regarding the locations of turtles 2 and 5, and opponents 3 and 4. However, on the basis of the trajectories of the players over a certain time interval up to time t^*, shown in Figure 6, the outcome does make sense. This clearly illustrates the advantage of labeling combined with
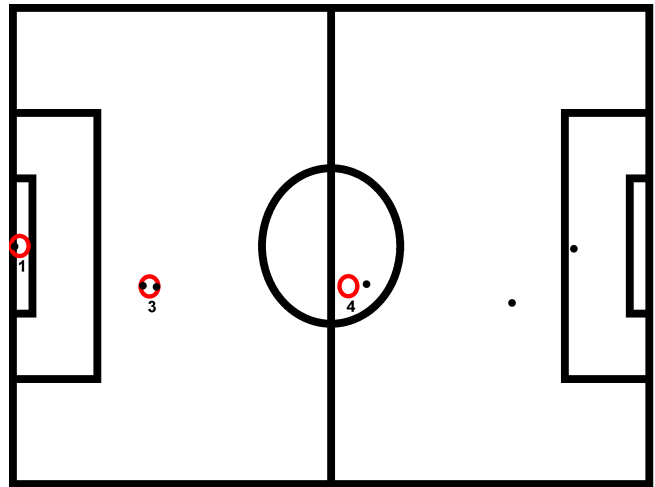

Fig. 3.   Local world model output from turtles 1, 3, and 4 (represented by black dots) and turtles locations (represented by red circles).
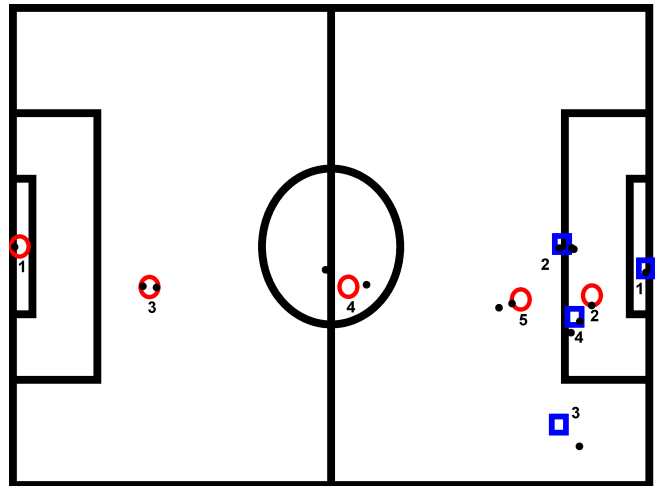

Fig. 4.   Output of all local world models (represented by black dots) and the output of the global world model. Turtle and opponent locations are represented by red circles, respectively blue squares.

tracking, *i.e.*, it can be avoided that tracks are lost during path crossing or temporarily occlusion and a more reliable output is generated as a result of the improved robustness.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper, a two level algorithm for developing a real-time world model that can be used for cooperating robots is developed. The local low level world model effectively reduces the amount of measurement data such that the global high level world model can run in real-time. The
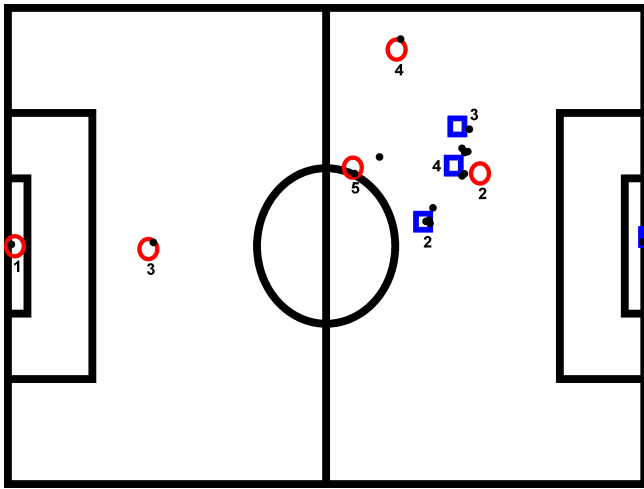
Fig. 5. Output of all local world models (represented by black dots). Turtle and opponent locations determined by the global world model are represented by red circles, respectively blue squares.
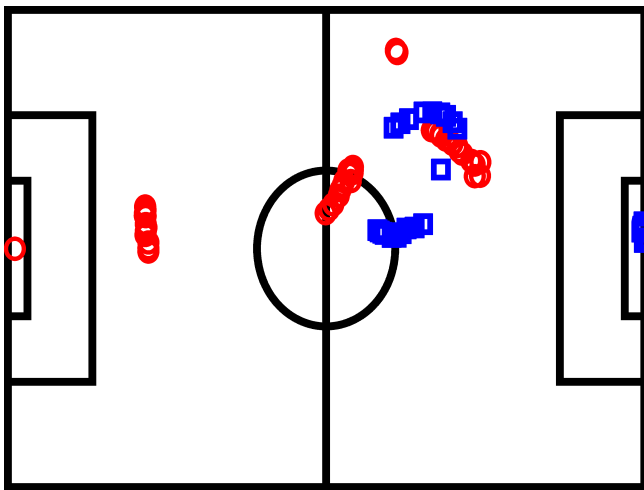

Fig. 6. Global world model output over a certain time interval up to time $t*$. Turtles are represented by red circles, opponents by blue squares.

low level runs at each robot and is used for local tasks such as obstacle avoidance, whereas the high level represents a consistent view of the world to all cooperating robots using the output of their low level world models. One of the main benefits of this approach is that all robots have the same information, which allows for easy and effective implementation of strategies. Furthermore, the global world model covers a larger area than the local world models.

This paper uses a Bayesian MHF with constant gain Kalman filters for target tracking. In addition, the contributions of this paper are:

- A successful extension of the filter with a heuristic labeling strategy
- An extension to a two level approach which allows real-time implementation
- A real-time demonstration of the algorithm that was presented with a team of robots

B. Future Works

In this paper, simple constant gain filters are used for object tracking. It is expected that a more advanced strategy for determining these gains will further improve the result. Also, the use of better probabilistic models for (4) – (6) can improve the result. Experimentally obtained data can be used to obtain such improved models.

The results in this paper clearly show the power of sharing knowledge among multiple robots. This idea is the basis of the RoboEarth project, see [12], where any useful knowledge obtained by a particular robot will be stored in a world-wide-web style database such that other robots can use it to improve their performance. In other work, the focus will be on the further development of the RoboEarth approach.

REFERENCES

[1] T. De Laet, Rigorously Bayesian Multitarget Tracking and Localization, *Ph. D. thesis*, Katholieke Universiteit Leuven, 2010. ISBN: 978-94-6018-209-9
[2] D.B. Reid, An algorithm for tracking multiple targets, *in IEEE Trans. on Automatic Contr.*, vol. **24**(6), 1979, pp 843–854.
[3] S. Chang, R. Sharan, M. Wolf, N. Mitsumoto and J. W. Burdick, People Tracking with UWB Radar Using a Multiple-Hypothesis Tracking of Clusters (MHTC) Method , *in Int. J. of Social Robotics*, vol. **2**(1), 2010, pp 3–18.
[4] S.S. Blackman, Multiple Hypothesis Tracking For Multiple Target Tracking, *in IEEE Aerospace and Elec. Systems Magazine*, vol. **19**(1), 2004, pp 5–18.
[5] J. Vermaak, S.J. Godsill, and P. Pérez, A Monte Carlo Filtering for Multi-Target Tracking and Data Association, *in IEEE Trans. on Aerospace and Elec. Systems*, vol. **41**(1), 2005, pp 309–332.
[6] D. Schultz, W. Burgard, D. Fox, and A.B. Cremers, People Tracking with Mobile Robots Using Sample-based Joint Probabilistic Data Association Filters, *in The Int. J. of Robotics Research*, vol. **22**(2), 2003, pp 99–116.
[7] Y. Bar-Shalom and T.E. Fortmann, *Tracking and Data Association*, Academic Press, 1988.
[8] R.P.S. Mahler, Multitarget Bayes Filtering via First-Order Multitarget Moments, *in IEEE Trans. on Aerospace and Elec. Systems*, vol. **39**(4), 2003, pp 1152–1178.
[9] R.P.S. Mahler, PHD Filters of Higher Order in Target Number, *in IEEE Trans. on Aerospace and Elec. Systems*, vol. **43**(4), 2007, pp 1523–1543.
[10] B.T. Vo, Random Finite Sets in Multi-Object Filtering, *Ph. D. thesis*, School of Electrical, Elec. and Computer Engineering, The University of Western Australia, 2008.
[11] J. Schubert and H. Sidenbladh, Sequential clustering with particle filters - Estimating the number of clusters from data, *in Proc. Eighth Int. Conf. Information Fusion (Fusion 2005)*, 2005, pp 1–8.
[12] O. Zweigle, M.J.G. van de Molengraft, R. d'Andrea and K. Häussermann, RoboEarth: connecting robots worldwide, *in ICIS '09: Proc. of the 2nd Int. Conf. on Interaction Sciences*, 2009, pp 184–191.