Using Particle Filters to Anticipate the Location of Reappearance of a Temporarily Hidden Target

Achim Lewandowski* achim.lewandowski@ofai.at

Patrick M. Poelz* patrick.poelz@ofai.at

Georg Dorffner** Georg.Dorffner@Meduniwien.ac.at

*Austrian Research Institute for Artificial Intelligence, Vienna
** Department of Medical Cybernetics and Artificial Intelligence, Center for Brain Research, Medical University of Vienna, Austria

Abstract

A robot and a moving target act in a known environment. Obstacles, predominantly walls, may hinder free view and motion. The task of the robot is to anticipate a possible location of reappearance, if the target is currently hidden. We assume in our approach that the belief about the current state of the target can be expressed as a time-dependent probability function, which itself is represented by means of particle filters. The robot knows the dynamics of the target or is at least allowed, during a training phase, to learn the one step forecast of the state of the target, including the behavior near to a wall. We show how to forecast the state of the target, how to use the robot's observations for an update of the current belief, and finally how to take advantage of the approach by predicting a possible location of reappearance to which the robot might be sent.

1. Introduction and general approach

Consider the task that a mobile robot is supposed to track a moving object within a given environment. Walls cause the target to be out of sight even for longer time periods. Our main interest is then to provide the robot with anticipatory capabilities to predict the location at which the target probably will reappear. These predictions can then be used to actually catch the moving object. Although strongly related to the task of simply estimating the location, the latter task requires additional knowledge about time passed until the forecast reappearance and about the distance between the robot and location of reappearance. The robot must assume that it can reach the chosen location before the target, assuming that the target will reappear at this location at all.

We would like to simplify the task and always assume that the robot is faster than the target, and even if it will miss the target, will be usually in a better position for the next attempt. We are more interested in modeling an anticipatory behavior by forecasting the location of reappearance. The concrete exploitation of this knowledge by judging whether an attempt will be successful is a second, related problem. We would like to model a behavior that is more sophisticated than simple reactive target following, which only works when the target is visible. The desired behavior should also be more intelligent than the level of reactive behavior, in which the robot always immediately follows the *assumed* location of the target. Figure 1 illustrates the two approaches and our new approach presented in this paper for the simple case of a single wall. While the first approach will fail, as the robot does not know what to do, when the target is out of sight, the other two approaches succeed. The approach we would like to present in this paper is built on a forecast, anticipated situation. Nonetheless the presented model will be less sophisticated than a model capable of finding a hidden motionless ball. As we will see later, our approach can be interpreted as reactive behavior in a forecast scenario.

We assume that, given the map of the environment, the motion dynamics of the target are fully described by its current location and its velocity vector. Furthermore we assume that the behavior is local in the sense that only a wall which could be reached during the next time step may have an impact onto the future trajectory. We assume some canonical form implicitly using the distance and the angle under which the target is moving with respect to the wall. The behavior will be identical for all cases, for which the robot approaches a wall under a certain angle, no matter, where on the map the wall is exactly located. This behavior does not need to be deterministic, but the probability law is assumed

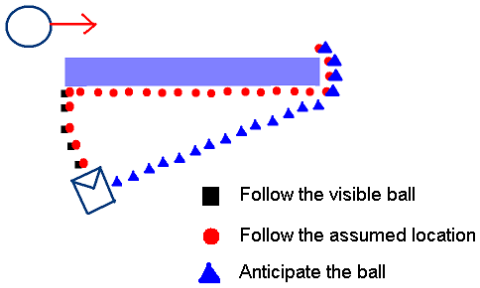

Figure 1: Three more or less successful approaches of catching the target.

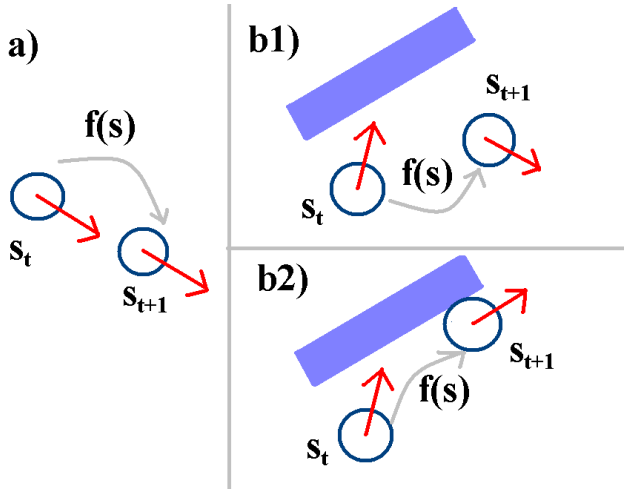

Figure 2: From state to state: a) Possible relationship between two consecutive states b1) and b2) Alternative behaviors near walls.

to be identical for identical situations. Similarly, the robot will always act identically in the open field.

Usually, we assume simple reflection at walls as in the case of a ball. Alternatively, completely different assumptions for the behavior near walls are possible, as e.g. "move parallel to the wall, with the wall to the left hand", or "turn 90 degrees clockwise, and continue with the same absolute velocity".

The target does not take notice of the robot and will not adapt its behavior depending on actions of the robot. The behavior of the target can therefore be completely described by the following model. If $s_t = [x_t, v_t]$ denotes the current state of the target, with $x_t \in \mathbf{R}^2$ denoting the location and $v_t \in \mathbf{R}^2$ denoting the velocity vector, we assume that the knowledge of the probability function

$$p(s_{t+1}|s_t), \tag{1}$$

is sufficient to describe the motion model of the target. It should be impossible that a target "traverses" a wall. Sometimes it is also convenient to express s_{t+1} as $s_{t+1} = f(s_t) + e(s_t)$, with e denoting an error term which is allowed to depend on the state s_t.

Figure 2 shows examples for free movement and two alternative behaviors near walls. We require that the new location x_{t+1} cannot lie behind the wall.

In our framework, we suppose that the robot either knows the law from (1) or that the robot is allowed to observe the target behavior, in the sense that it can learn an approximate relationship between two consecutive states during a training phase. Therefore, the robot must be able to measure locations and velocities. It is also necessary to assume that the robot has a map of the environment. At the end of the training phase, a simulation model will have been generated, which tries to imitate the one step dynamics of the target:

$$p^*(s_{t+1}|s_t), \tag{2}$$

Again we might use the form $s_{t+1} = f^*(s_t) + e^*(s_t)$. By repeated application of one of the models (1) or (2) a complete trajectory of the target can be simulated.

No matter how complicated the assumptions about the moving behavior of the target are, we could always express, at least in principle, the current belief about the state of the target with the aid of a probability function. The explicit handling of this function can be considered as feasible in special cases, e.g. if no or only very few walls were present. In our more complicated scenarios, use of exact derivations seems to be prohibitive. Thus we prefer to use particle filters.

Particle filters are becoming more and more popular in the robotics community. Thrun et. al mention on page 437 "that particle filters are at the core of some of the most effective robotics algorithms" (Thrun et al., 2005). In chapter 13 particle filters are used to treat the SLAM (Simultaneous Localization and Mapping) problem, but there the robot needs to localize *itself*. Approaches using particle filters for target tracking have already been proposed: Schulz et al. developed a hybrid version using particle filters to simultaneously predict the trajectories of multiple (but visible) objects (Schulz et al., 2001). Yu and Cheng use particle filters to track visible objects on roads (Yu and Cheng, 2006). Their approach tries to cope with the specific problems of sudden changes in velocity and the ambiguity problems at intersections.

A similar problem is tackled by (Kirubarajan et al., 1998) with a VS-IMM (Variable Structure Interacting Multiple Model) approach.

Their targets are vehicles on a highway which are observed from above by aircrafts. Although a percentage of the objects is allowed to leave the roads, the vehicles are generally assumed to stick to the road network. Visibility conditions are allowed to change.

Herman et al. present another particle filter algorithm to perform joint multitarget tracking (Herman and Roberts, 2005). Hue et al. also use particle filters for multitarget tracking, with the emphasis put on the problem of data association (Hue et al., 2000).

The approach used by (Mottaghi and Vaughan, 2006) and (Mottaghi and Vaughan, 2007) is similar in spirit to ours. They assume that an intruder should be tracked down. The current position of the intruder is approximated by a sample of particles. Mottaghi et al. put the emphasis on the cooperation of several robots. Their dynamic model to extrapolate particle positions differs from our model in the point that in our approach particles are not allowed to traverse walls. Our model uses in some points similar, but in certain aspects quite different updates of the particle sample when using the robot's observation. The model from Mottaghi et al. uses the *current* position of particles, whereas our approach also use *forecast* positions. Their model differs also in the way, how the direction to which the robot(s) are supposed to proceed is generated.

In the next section 2. we will give a short introduction how particle filters are incorporated in our framework. The following section 3. deals with the process of deciding between different directions the robot could choose. Then, in section 4., we show the qualitative results of some simulation experiments. In the final section 5., we summarize our results and mention ideas for possible extensions of the approach.

2. Particle filter

Applying a particle filter is a nonparametric implementation of the Bayes filter (for an elaborate description see e.g. (Thrun et al., 2005)). The object of interest is the current belief about the state of the target, expressed by a probability distribution over location and velocity. As time passes, the probability distribution of the state of an unobserved target becomes more and more complicated. The presence of obstacles generates multimodal distributions, as the target might have taken either of both ways to pass the obstacle. Particle filters approximate this complicated posterior distribution of the state by a sample of particles distributed according to this distribution.

Assume that at time $t-1$ a sample of particles $s_{t-1}^{[m]}$, $m = 1, \ldots, M$, is given. Each particle can be seen as a possible state of the target at time $t-1$. During $t-1$ and t each particle changes according to the probability function $p(s_t^{[m]}|s_{t-1}^{[m]})$ from (1). $p(s_t^{[m]}|s_{t-1}^{[m]})$ describes the probability of a target being in state $s_t^{[m]}$, assuming that the last state was given by $s_{t-1}^{[m]}$. The control variable u_t which usually expresses the performed action is already incorporated, as the target always behaves identically under identical circumstances. Actions of the robot do not change the behavior of the target and do not need to be considered.

As the dynamic model used by the robot is expressed by one of the equations (1) or (2), we now turn to the question how to incorporate the observations of the robot, that is how to calculate the weights of the importance sampling, the so-called importance factors.

We assume that only the locations of the robot and the particle should have a significant influence on the value of a weight.

The possible scenarios are the following:

1. The robot could not detect the target.
2. The robot has detected the target at position z.

We assume that the robot only delivers an estimate of the location of the target, if the target is visible. Therefore we do not have any measurement of the location, if the target is hidden behind a wall. We also assume that the robot always detects a visible target, and delivers an estimation z of the exact location. For simplicity reasons we assume that the robot has a certain viewing cone ((Mottaghi and Vaughan, 2006) additionally assume that the range of the sensor is restricted) and in the case that the robot cannot detect the target, every particle inside this cone gets a low weight $c \geq 0$. All particles which are hidden behind a wall in comparison get a higher weight $d > c$, as they do not stand in contradiction to the target-free viewing cone. Under the assumption that the robot always notices a visible target, c could be set to 0, in which case all visible particles will be removed immediately, if the target is invisible. Using a low value of c ($0 < c \ll d$) will also erase particles from the sample in a small number of steps and mitigates the problem of assigning weights when all particles are visible, but the target is not.

In the case that the target is visible, we assume that the weight depends only on the distance $D(x_t^{[m]}, z)$ between location measurement z and the location $x_t^{[m]}$. We suggest to assign weights as e.g. in

$$w(s_t^{[m]}|z) = \exp\{-a\,[D(x_t^{[m]}, z)]^2\} \qquad (3)$$

with a constant $a > 0$ describing the quality of the vision system.

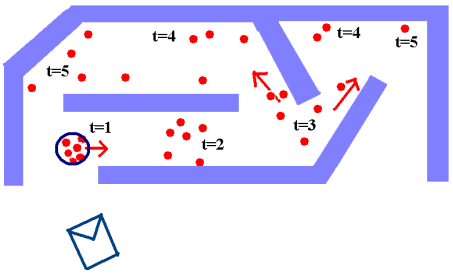

Figure 3: Example of assumed locations of particles, if target moves behind walls.

In the case of a visible target, the formula from (3) is applied both to hidden and visible particles. Assigning different values to hidden particles (e.g. $w = 0$ or $w = c$ for hidden particles, as their probability for representing the true location is 0, otherwise the target would not have been seen) can be done, but does not seem to be necessary. A sample of particles is drawn now (with replacement) with probabilities being proportional to the weights.

Using the described procedure, those particles that are in accordance with the observation of the robot tend to remain in the sample. Strictly speaking, we should have used also the similarity between particle velocity and measured velocity to calculate the weights. If we define smooth motion as a motion for which the target's behavior depends continuously on location and velocity and therefore small changes in one of the variables do not generate sudden changes in the behavior of the target, consideration of the velocity is not necessary. It *might* happen that a particle going to the opposite direction will be near to the measurement z and will receive a higher weight, but this is only a problem, if it is the single particle which remains in the sample or if the target is observed only for one time step. If the target can be observed at least twice, particles with the "wrong" velocity vector will depart from the target and therefore the probability to remain in the sample becomes smaller for them.

The robot always has a certain belief about the position of the target. The cloud of particles tends to be relatively narrow (depending on the quality of sensors), if the target is visible, and starts to grow for an invisible target. Figure 3 shows the possible evolution of a particle cloud, after the target - in this case a ball - has vanished behind the walls.

3. Motion of the robot

We assume that the main task for the robot is not to hunt a visible target, but to anticipate the next appearance, under the condition that walls often do not allow an unobstructed view of the target. The decision, whether the robot will be able to catch the target at the forecast position depends on the distance from the robot to this point and on the forecast time span the target might need to reappear. As we assume that the robot can measure the velocity of the target and knows its own velocity, we can calculate the time needed to reach this point and also determine, whether the robot will be sufficiently fast to reach the target.

The robot only "thinks" in particles. Actually observing the target gives the possibility to improve the current quality of the particle representation of the true state. Any intended motion just relies on the particle locations. We assume a reactive behavior in the sense that the robot will try to follow particles if they are visible. Here, "visible" means that a target with the same location could be seen by the robot. Particles in the back of the robot are not visible.

We have several options:

1. The robot follows the gravity center of all visible particles. This seems to be a good idea, if the particles form only one cluster, but the robot might get stuck, if the particle cloud consists of two or more diverging clusters.
2. The robot follows a randomly chosen visible particle.
3. The visible particles could be clustered in advance, and the robot follows that cluster center, which belongs to a randomly chosen visible particle or contains the most particles.
4. The robot follows the nearest visible particle.

During our experiments, we have found that strategy 4 usually works reliably.

If the robot follows any visible particle and the target does not appear, then the visible particles will most likely vanish after a short time, as their weights are 0 or at least lower than the weights of hidden particles. In this case there will be no visible particle to follow, but we can still work with the hidden particles. We assume that the robot has an internal representation of all particles. In this internal representation it also knows the location of particles which are in reality behind its back.

In the case that the robot cannot "see" any particles, two types of particles exist. The first type are particles hidden behind the walls and the second type consists of particles which are behind the back of the robot. The view of these particles is not blocked by any walls.

In its internal representation, the robot knows the particles of the second type, and therefore the robot turns and moves towards available particles in its back. But after a short time, even those particles, which, after turning, transform into "visible" particles will disappear, if the target could not be found. Now, only particles hidden behind walls exist. In a very complicated map and assuming a small view cone it is imaginable that always new particles will arrive in the back of the robot, and then the robot would consistently turn around without ever sweeping away all particles. But we assume now that the robot has arrived at a state where all particles are hidden behind walls.

We assume that the robot in its internal representation starts to simulate future trajectories of all hidden particles according to equation 1. Possible guidelines are to stop this procedure, if e.g.

1. at least one of the extrapolated hidden particles will reappear or

2. all of the hidden particles will reappear.

The bottom image of figure 5 shows the extrapolated trajectories according to case 1.

In either case, the robot can treat the extrapolated particles as in the case of immediately visible particles as described above and might move, for instance, to the nearest extrapolated particle. This approach can be seen as reactive behaviour in a forecast scenario. The robot does not have the capacity to find a hidden particle, even if it knows the location, but it is able to extrapolate the particle locations until they are visible and can react to these imagined visible particles. Whereas the robot of Mottaghi et al. will always choose a direction which is supposed to be optimal for the *current* location of the target (Mottaghi and Vaughan, 2006), our robot tries to anticipate future behaviour. It should be noted that our approach does not guarantee that our robot will find the target earlier than the other approaches.

The number of particles must be increased, if the robot has to move in a complicated map with many obstacles. The reason is that for every obstacle a particle can take one of two ways, and the number of possible paths increases rapidly. If the number of particles is too low, it can happen that no particle is near to the true position of the target. This is a lesser problem, if the target can be observed for a longer time, as those particles, which are nearer to the target, tend to survive. If the target can only be seen for a short time, even the resampled particles will still be quite far away, and the robot will follow particles without ever finding the target. A large number of particles prevents this scenario, as with higher probability at least some of the particles are approximately behaving like the target. If we work with a large number and all particles are currently invisible behind a long wall, we will have the problem of simulating long trajectories of many particles.

We developed two strategies to cope with the mentioned problems. The case that no particle is nearer to the visible target than a given radius r can be detected by simply noting that none of the weights according to equation (3) exceeds the value $\exp[-ar^2]$. This scenario can be labeled as "surprise", as none of the particles fit well to the measurement, and the strategy we use is to start with a new sample of particles all set to the current measured state.

The other problem of long forecast horizons can be somehow relaxed. Usually the whole sample of particles is extrapolated until at least one of the particles reappears. The simple solution to this problem is to perform these calculations only for a randomly chosen smaller subsample.

The second approach performs the time-consuming simulations only once and lets the robot first move to the extrapolated position, before any new simulated trajectories are calculated. This procedure is interrupted, if the target has been sighted before the robot has reached the extrapolated position.

4. Experiments in simulated worlds

We constructed several artificial worlds and started with a ball moving behind a single wall. The ball physics was chosen according to the law "angle of reflection equals angle of incidence", including some noise. The accuracy of the vision system was assumed to be high, which is equivalent to rapidly decreasing weights by increasing distances between particles and target. Figure 4 illustrates the different phases which occur. As long as the robot can observe the ball, the particle cloud tends to be small (a). If the ball vanishes, there might be some visible particles at the left edge to which the robot could walk to, but in this example they disappear within a few steps (b). If all particles are invisible (c), the robot moves immediately to the nearest visible *extrapolated* particle, which happens to appear at the right end. The size of the cloud grows (d). When the first particles become visible at the right end, the robot moves to the nearest of them. As long as the ball is not visible, these particles tend to disappear (e). Finally, the ball reappears, and the particle cloud shrinks again.

We tested the algorithm on more and more complicated maps, starting with a map with circular arcs (see figure 5), for which the usual ball physics can be applied, if circle arcs are locally approximated by tangents. In figure 5, the ball is placed with a random velocity vector parallel to the tangent, whereas the robot is placed inside the inner circle. Figure 5 shows a run, for which the robot changes its direction several times.

Finally, we used an example with many walls (see

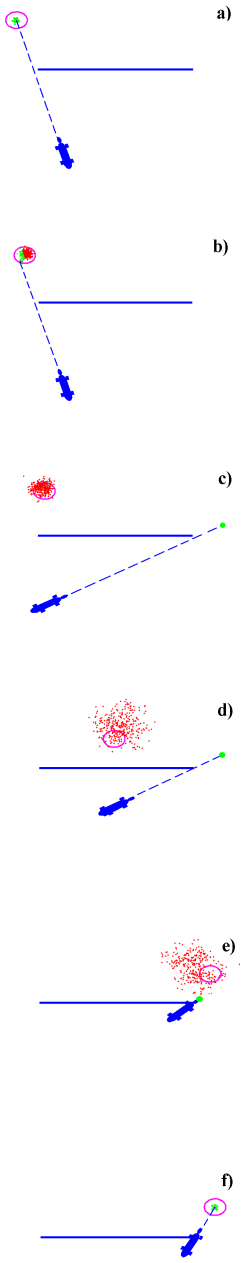

Figure 4: The robot observes the ball rolling behind the wall. a) The robot can see the ball, the particle cloud is small. b) The ball vanishes behind the wall. Some particles are still visible. The cloud begins to grow. c) All particles are invisible now. The robot moves to the first visible extrapolated particle which in this example happens to appear at the right end. d) The particle cloud grows further. The robot moves to an extrapolated particle appearing at the right end. e) First particles are visible. The robot moves to the nearest of them. f) Ball is visible now. The robot moves to the nearest particle in the cloud.

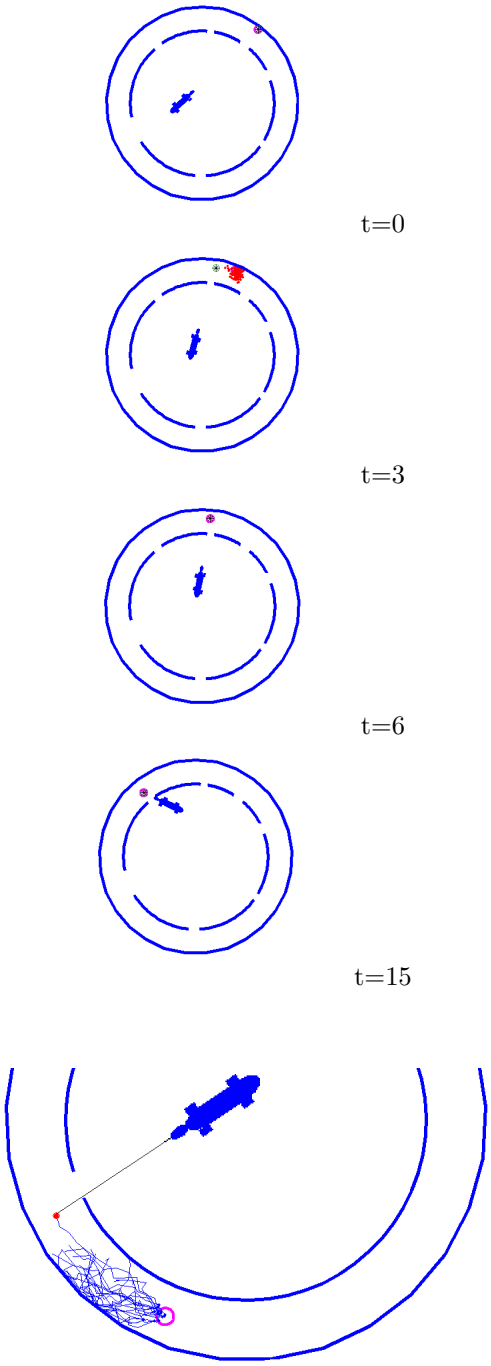

Figure 5: The robot must find a moving ball in an environment with circular walls. The ball is placed randomly in the outer ring. The robot is placed inside the inner circle. The robot approaches the ball, but in t=3 all particles are invisible and the robot changes the direction. In t=6 the ball is visible for a short time, but vanishes immediately. The robot needs to change the direction once again. The bottom image shows the simulated trajectories of hidden particles and the robot will move to the first "visible" particle.

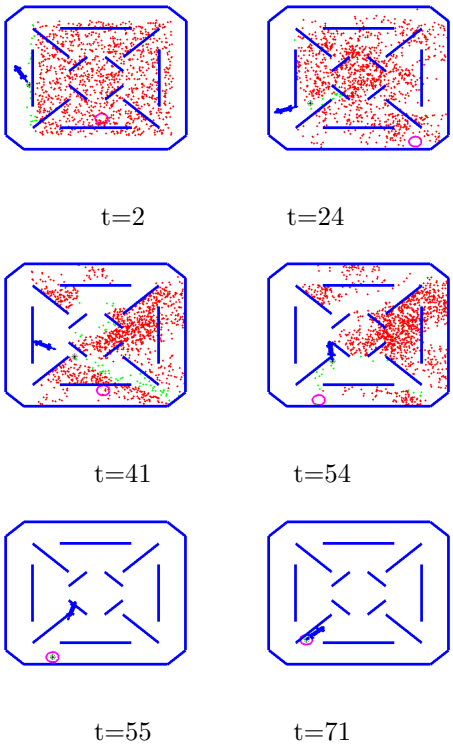

Figure 6: The robot must find a moving ball in an environment with many walls. The ball is placed randomly in a smaller rectangle with a random velocity vector. The first particle sample is generated according to the same law. t=2) Many particles arrive and the robot cannot advance significantly. t=24) The nearest particle is located further to the right and the robot leaves the area it started in. t=41)t=54) The robot sweeps away the particles, but it does not observe the target. t=55) The robot observes the target. t=71) The robot catches the target.

figure 6). The robot does not know the starting position of the target, which was randomly generated inside a rectangular area nearly as large as the whole map. The first sample of particles was generated using the same law. During the first time steps, the robot cannot decide to which end of the neighboring wall it should go, as constantly new particles arrive (and vanish as the target is not visible). After it has managed to get around the wall, it sweeps away all visible particles, until it finally observes and catches the target.

In our simulations, the robot nearly always found the target. Only in very complicated maps and if the robot is equipped only with a small viewing angle, the robot might need very long to sweep away the ever-arriving particles in its back.

5. Conclusion and Outlook

We developed our algorithm with the goal to implement a behaviour which uses anticipation, but combined with some reactive behavior. Therefore our robot will only find targets which are or will be visible from the current viewpoint (or from viewpoints which might be generated while the robot is moving). Our robot, equipped with our proposed algorithm, does so far not have the capacity to find a target with known location which stopped behind a wall. The robot *might* find the target nonetheless in the case that particles will reach the ends of the wall often enough to draw the robot to one of both ends, from which it might see the target itself.

We assume in (1) a dynamic model of a target that is always in motion, e.g. in the simplest case a target that moves with constant absolute velocity. The model used by the robot is either identical to this model or an approximation developed during the learning phase. This means that apart from exceptions all particles will move endlessly and, under mild assumptions, will appear in every part of the map with a positive probability. We just need to wait sufficiently long to see each particle again.

If we consider the case for which the absolute velocity is allowed to decrease, e.g. due to friction, until it reaches zero, then the particles will come eventually to a stop. If the friction is so high that not a single from the hidden particles will reappear behind the wall, our algorithm cannot be applied.

If in addition we assume that the robot has an algorithm implemented to reach each given coordinate, visible or hidden, we can extend our algorithm: During the extrapolation part, for each forecast horizon a location coordinate is calculated from the simulated particle trajectories. With help of the additional algorithm, a path is constructed and the length of this path is calculated. If the robot is sufficiently fast to reach the location within the given time, the direction along the path is chosen. Of course, even more sophisticated algorithms can be developed, e.g. algorithms which maximize the probability to catch the target. The simulated particle trajectories could be helpful for this task, as they might be used to estimate probabilities of having a distance $d \leq k$ to the true position of the target.

We are currently transfering our algorithm to a real robot. Another simpler robot will act as a target moving according to some simple rules. The success of our approach will mainly depend on the ability of the main robot to localize itself and the second robot.

Acknowledgements

This research has been done within the project "MindRACES", funded by the European Commu-

nity in the "Competitive and Sustainable Growth Programme", under contract number IST-511931.

One of the authors has received financial support from the SELP program.

The Austrian Research Institute for Artificial Intelligence acknowledges basic financial support by the Austrian Federal Ministry for Education, Science, and Culture and of Transport, Innovation and Technology.

References

Herman, S. M. and Roberts, S. E. (2005). Efficient multitarget particle filters for ground target tracking. *Proceedings of the SPIE. Signal and Data Processing of Small Targets 2005. Edited by Drummond, Oliver E.*, 5913:254–268.

Hue, C., Cadre, J. L., and Perez, P. (2000). Tracking multiple objects with particle filtering. *Technical report, IRISA (Institut de recherche en informatique et systèmes aléatoires).*

Kirubarajan, T., Bar-Shalom, Y., Pattipati, K. R., and Kadar, I. (1998). Ground-target tracking with topography-based variable-structure IMM estimator. *Proceedings of the SPIE. Signal and Data Processing of Small Targets 1998. Edited by Drummond, Oliver E.*, 3373:222–233.

Mottaghi, R. and Vaughan, R. (2006). An integrated particle filter & potential field method for cooperative robot target tracking. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1342–1347.

Mottaghi, R. and Vaughan, R. (2007). An integrated particle filter and potential field method applied to cooperative multi-robot target tracking. *Autonomous Robots (to appear!).*

Schulz, D., Burgard, W., Fox, D., and Cremers, A. (2001). Tracking multiple moving objects with a mobile robot using particle filters and statistical data association. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1665–1670.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press.

Yu, Y. and Cheng, Q. (2006). Particle filters for maneuvering target tracking problem. *Signal Process.*, 86(1):195–203.