Stable Mapping Using a Hyper Particle Filter

Johannes Pellenz and Dietrich Paulus

Active Vision Group, University of Koblenz-Landau,
Universitätsstr. 1, 56070 Koblenz, Germany
{pellenz, paulus}@uni-koblenz.de

Abstract. Often Particle Filters are used to solve the SLAM (Simultaneous Localization and Mapping) problem in robotics: The particles represent the possible poses of the robot, and their weight is determined by checking if the sensor readings are consistent with the so far acquired map. Mostly a single map is maintained during the exploration, and only with Rao-Blackwellized Particle Filters each particle carries its own map. In this contribution, we propose a Hyper Particle Filter (HPF) – a Particle Filter of Particle Filters – for solving the SLAM problem in unstructured environments. Each particle of the HPF contains a standard Particle Filter (with a map and a set particles, that model the belief of the robot pose in this particular map). To measure the weight of a particle in the HPF, we developed two map quality measures that can be calculated automatically and do not rely on a ground truth map: The first map quality measure determines the contrast of the occupancy map. If the map has a high contrast, it is likely that the pose of the robot was always determined correctly before the map was updated, which finally leads to an overall consistent map. The second map quality measure determines the distribution of the orientation of wall pixels calculated by the Sobel operator. Using the model of a rectangular overall structure, slight but systematic errors in the map can be detected. Using the two measures, broken maps can automatically be detected. The corresponding particle is then more likely to be replaced by a particle with a better map within the HPF.
We implemented the approach on our robot "Robbie 12", which will be used in the RoboCup Rescue league in 2009. We tested the HPF using the log files from last years RoboCup Rescue autonomy final, and with new data of a larger building. The quality of the generated maps outperformed our last years (league's best) maps. With the data acquired in the larger structure, Robbie was able to close loops in the map. Due to a highly efficient implementation, the algorithm still runs online during the autonomous exploration.

1 Introduction

Many different algorithms have been proposed in the last few years to solve the SLAM (Simultaneous Localization and Mapping) problem ([2–4, 6, 8]). These solutions nowadays run in real time (for 2D maps) and produce very accurate maps, which can be used by robots for path planning and navigation. To handle the uncertainties that result from noisy sensor data, probabilistic approaches became very popular within the last couple of years (see [9] for a survey). Within the probabilistic approaches, Particle Filters [10] have some favorable properties: They can handle the so called “kidnapped robot problem”, where the pose of the robot is unknown in the beginning. Also, multiple hypothesis of the robot pose can be tracked, and non-Gaussian distributions can be modeled.

However, the original implementation of Particle Filters suffered from the fact that only a single map (often as an occupancy grid) was maintained, and the particles only represented different assumed poses of the robot. So if the map “broke”, the complete map building failed. Rao-Blackwellized Particle Filters [1] can maintain a single map for each particle. So if a map breaks, the corresponding particle is finally removed. Hähnel et al. combined Rao-Blackwellized Particle Filtering and scan matching in the FastSLAM algorithm [5]. Here each Particle Filter contains a map as a list of features that are stored within an Extended Kalman Filter.

In this work, we propose a Hyper Particle Filter (HPF) – a Particle Filter of Particle Filters – for solving the SLAM problem in unstructured environments. Each particle of the HPF contains a standard SLAM Particle Filter: The particle contains a map and a set of particles that model the belief of the robot pose in this particular map. The Particle Filter requires a measurement step. To measure the weight of a particle in the HPF, we developed two *map quality* measures: The first map quality measure mq_1 determines the contrast of the occupancy map. This is motivated by the fact that if the map has a high contrast, it is likely that the pose of the robot was determined correctly before the map was updated. This finally leads to an overall correct map. The second map quality measure mq_2 determines the distribution of the orientation of wall pixels calculated by the Sobel operator. In contrast to mq_1, the measure mq_2 can only be used if the overall structure of the mapped area is known to be rectangular. Using this measure, slight but systematic errors in the map can be detected. Both quality measures can be calculated automatically and do not rely on a ground truth map. Using mq_1 and mq_2, broken maps can automatically be identified. The corresponding particle is then more likely to be replaced by a particle with a better map within the HPF.

The paper is organized as follows: Section 2 reviews the use of Particle Filters for robotic mapping and localization, section 3 introduces the concept of the Hyper Particle Filter. Section 4 describes experiments with the HPF. Section 5 concludes the paper, and section 6 closes the paper with the topics that are addressed in the future.

2 Particle Filters

The idea of Particle Filters is adopted from the field of computer vision, where the principle is known as the Condensation Algorithm [7]. The idea is to represent the distribution function of the robot pose as a set of particles: The more likely a particular pose is, the more particles represent this area. Whenever new sensor data is available, the following steps are executed (adapted to the field of robotics):

1. **Resampling**: A new generation of particles is created from the current set of particles. The higher the weight of a particle is, the more likely it is drawn and its pose is represented in the next generation.
2. **Drift**: The particles are moved according to the control update or the odometry readings of the robot. Additionally, a scan matcher is used to improve the estimated transformation.
3. **Diffuse**: The particles are moved according to the noise of the motion model.
4. **Measure**: Using the current sensor data (e. g. readings from the laser range finder) a weight is assigned to each particle (which represents the consistency of the pose and sensor data with the so far acquired map).

For our robot "Robbie X", we used a Particle Filter with about 1,000 particles that represent possible poses $(x, y, \Theta)^T$ in 2D. Note that the map itself is *not* part of the state vector.

In our approach, the Particle Filter is used for the localization only: The acquired map is stored independently in an occupancy grid [3]. In the context of RoboCup Rescue, the grid has a size of 800×800 cells, which represents a map of an area of 40×40 meters with a grid cell size of 50×50 millimeters. The grid is stored in two planes: One plane counts how often a cell was "touched" by a laser beam. This value is increased either if the laser beam reported the cell as free or if the cell was reported as occupied. A second plane stores the information how often a cell was seen as occupied. By dividing the values of these two planes, the occupancy probability for a cell c_i is estimated by the following ratio:

$$p_{\text{occ}}(c_i) = \frac{\text{count}_{\text{occ}}(\text{c}_\text{i})}{\text{count}_{\text{seen}}(\text{c}_\text{i})} \tag{1}$$

To extend the map, the best pose that the Particle Filter determines is used to update the map: The current laser range scan is added to the global occupancy map at the estimated robot pose by constructing a local map and "stamping" it into the global map, incrementing the counts for *seen* and *occupied* cells. Special attention is paid to cells that are touched by the laser beams more than once during a single scan (these are cells that are close to the robot): If such a cell is seen as occupied, than other beams of this scan can not overwrite this cell as free any more.

So far only a single map was maintained, and in case of a defect in the map, there was now way to recover.

3 Hyper Particle Filter (HPF)

To overcome the problem of broken maps, we use multiple (about 30) SLAM Particle Filters concurrently. Due to the probabilistic behavior of each filter in the diffusion step, different (but similar) maps are generated in each SLAM filter. Additionally, each filter can reject a new measurement with a 20% chance. This way, some of the filters are not influenced by totally broken scans, that might be the result of scans taken while the robot was turning or tilting on top of a ramp. This happens frequently at the rough environment that is simulated in the RoboCup Rescue arena. Even though we use a gimballed laser range finder, which is actively balanced using two servo motors and the data of an dual-axis accelerometer, we get distorted scans. This frequently happens when the robot drives over the top of the ramps.

The 30 SLAM Particle Filters are organized within another Particle Filter, the so called Hyper Particle Filter. This is depicted in Fig. 1. The particular steps of this Particle Filter are implemented as follows:

1. **Resampling**: During the resampling step, SLAM Particle Filters that carry a map with a weight below 99% of the weight of the map of the best SLAM Particle Filter are replaced by the best one (including map and robot poses).
2. **Drift**: During the drift step each SLAM Particle Filter is updated with the current laser scan.
3. **Diffuse**: (No diffusion.)
4. **Measure**: For the measurement step, the map quality of each SLAM Particle Filter is analyzed using the map quality measures mq_1 and mq_2 that are described below.

3.1 Map quality measure mq_1

The ability to generate high quality maps is strongly related to the ability to locate the robot when the range measurement was taken: If the localization was correct (and the noise of the sensor is reasonably low), then the resulting map shows consistent assumptions for walls and other features. With incorrect pose estimations, the assumptions for obstacles are not consistent, which results in blurry regions in the occupancy grid. These situations are illustrated in Fig. 2: Two scans are take from the same position, with a different orientation (see Fig. 2(a) and 2(b)). If the rotation between the two scans is estimated correctly, then the two scans can be registered perfectly (see Fig. 2(c)). If there is an error in the estimation, then the scans are merged incorrectly in an occupancy grid (see Fig. 2(d)): The occupancy probability for parts of the map cells drops, because the location that corresponds to these cells have been seen once as free and once as occupied. It is likely that this inncorrect registration is the starting point for a broken map.

The value of mq_1 measures the contrast of the map $\mathbf{M_{occ}}$ by checking for the absence of uncertain cells for all areas that have been scanned by the range sensor so far (these cells are in the set $\text{seen}(\mathbf{M_{occ}})$):

$$mq_1(\mathbf{M_{occ}}) = \frac{1}{|\text{seen}(\mathbf{M_{occ}})|} \sum_{c \in \text{seen}(\mathbf{M_{occ}})} \text{contrast}(\mathbf{M_{occ}}(c)) \tag{2}$$

with

$$\text{contrast}(\mathbf{M_{occ}}(c)) = \left(\frac{\mathbf{M_{occ}}(c) - 0,5}{0,5} \right)^2 \tag{3}$$

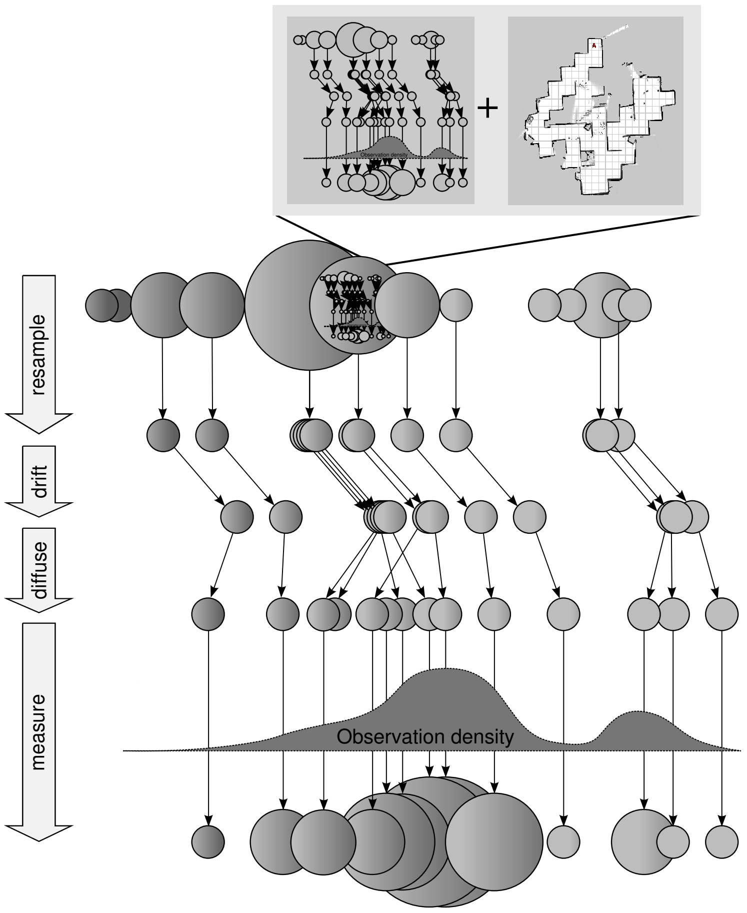
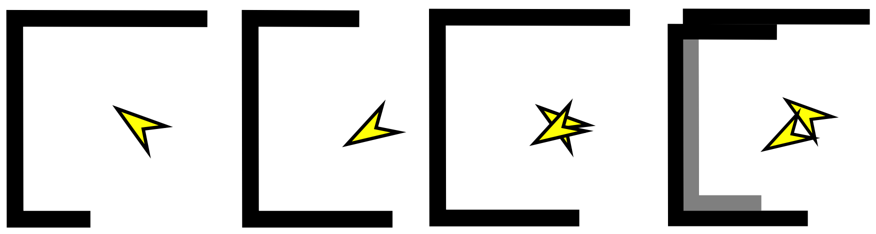


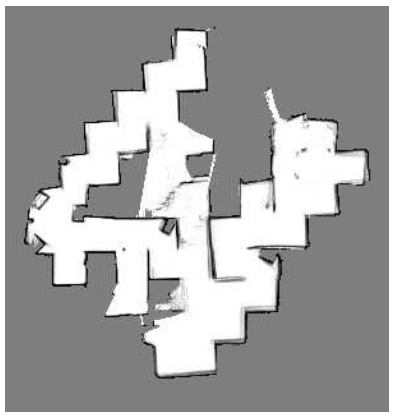
Fig. 1. The principle of the Hyper Particle Filter: The particles of the Hyper Particle Filter are standard SLAM Particle Filters (with a map and particles that represent robot poses). For the measurement step, the quality of each map is evaluated.



(a) First scan of the robot (b) Second scan of the robot (c) Correct registration (d) Incorrect registration

Fig. 2. Correct (c) and incorrect (d) registration of the two scans (a) and (b). The incorrect registration lowers the occupancy probability of some of the grid cells. This is an indicator for the incorrect pose estimation.

where $\mathbf{M_{occ}}$ is the occupancy grid representation of the map and $\mathbf{M_{occ}}(c)$ the occupancy value of a particular cell c of the grid. This way, a pixel which is clearly free ("white") or clearly occupied ("black") is counted with full contrast (= 1.0); a pixel which was scanned the same number of times free and occupied ("gray") is counted with no contrast (= 0.0). Fig. 3 gives an example for the measure mq_1 for two different maps. For the first map (Fig. 3(a)), only 50 particles were used; for the second map (Fig. 3(b)) 1000 particles. As expected, the value of mq_1 is higher for the map with more particles.


(a) Map generated with **50** particles: mq_1 = **87.39%**

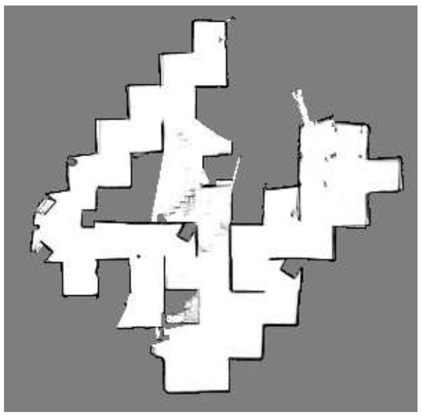

(b) Map generated with **1000** particles: mq_1 = **92.55%**

Fig. 3. Maps generated with different numbers of particles – and the corresponding map quality measure mq_1. The lower quality of map (a) compared to (b) is visually hard to detect, but sensed by the mq_1.

3.2 Map quality measure mq_2

The mq_2 measures the consistancy of wall directions within a map. Unlike mq_1, this measure is not model free: It can be used only if the building has a rectangular structure. The map becomes inconsistent if a systematic error causes the walls of the map to bend to a certain direction, or if the map is broken and the walls suddenly point to a random direction. An example of such a broken map is given in Fig. 4.

The directions of edges in an image can be calculated in various ways; one of the most common methods is the Sobel edge detector. Let $\mathbf{M_{occ}}$ be the occupancy grid of a building structure, and $\mathbf{S_x}$ and $\mathbf{S_y}$ Sobel operators for the x and y direction, then the horizontal and vertical derivative approximations $\mathbf{G_x}$ and $\mathbf{G_y}$ can be calculated as follows:

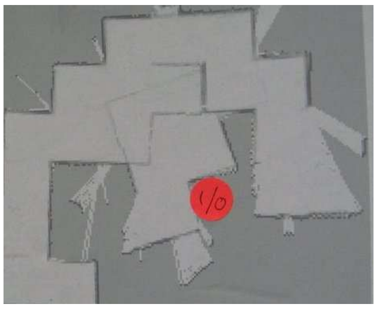


Fig. 4. Broken map: A part of the building (next to the round label) is turned and moved. Map of the autonomy final during RoboCup Rescue 2007, Atlanta, USA.

$$\mathbf{G_x} = \mathbf{S_x} * \mathbf{M_{occ}} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \mathbf{M_{occ}} \tag{4}$$

and

$$\mathbf{G_y} = \mathbf{S_y} * \mathbf{M_{occ}} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{M_{occ}} \tag{5}$$

where $*$ denotes the convolution operation. Using $\mathbf{G_x}$ and $\mathbf{G_y}$, we can calculate the strength of the edge and gradient's direction:

$$\mathbf{G}(\mathbf{M_{occ}}) = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2} \tag{6}$$

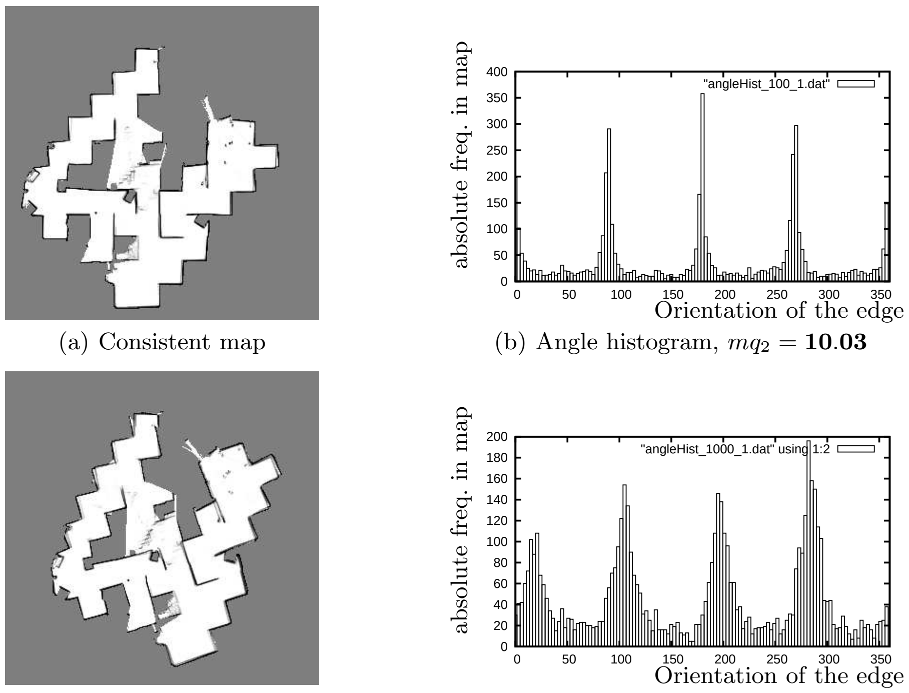
$$\mathbf{\Theta}(\mathbf{M_{occ}}) = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x) \tag{7}$$

Using the edge directions, we calculate the measure mq_2 for the consistancy of the wall direction using the following algorithm:

1. Calculate for a map $\mathbf{M_{occ}}$ the edge image $\mathbf{\Theta}(\mathbf{M_{occ}})$.
2. Create the histogram hist($\mathbf{\Theta}(\mathbf{M_{occ}})$) over the angles (bin size: 3°).
3. Smooth the histogram using a mean filter of size 5.
4. For the four largest peaks $p_i, i \in [1..4]$ in the histogram:
   Calculate μ_i, σ_i and $|p_i|$ (number of angles that belong to this peak). Use all angles within ± 30° around the peak.
5. Calculate mq_2: $mq_2 = \frac{\sum_{i=1}^{4} |p_i|\sigma_i}{\sum_{i=1}^{4} |p_i|}$

The value of mq_2 is small for significant peaks, and large for non-significant peaks. Fig. 5 gives examples for two different maps: Based on the same logfiles, the parameter that determines the distance between sampling points in the laser scan is modified. With a small distance between the sampling points (100 mm) a highly accurate map is produced (with the original, rectangular structure of the building). If the distance between the sampling points is increased (e. g. to

1,000 mm), the map gets inconsistent over time; resulting in skewed (and blurry) walls. Therefore, the orientation of the walls is not consistent any more, which is clearly visible in the histogram of the wall angles: The histogram is more flat, and therefore mq_2 – which adds up the variance of the four most significant peaks – is larger.



(a) Consistent map (b) Angle histogram, $mq_2 = \mathbf{10.03}$

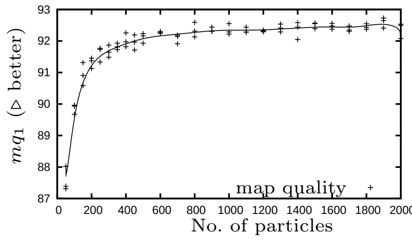
(c) Mostly consistent map (d) Angle histogram, $mq_2 = \mathbf{12.20}$

Fig. 5. Map and angle histogram to determine mq_2 (particle count = 5000, cell size 50 × 50 mm). In (a) ten times more distance measurements from each laser scan were used compared to (c). (data set: `rescueServer_2008.07.20_13-49-13.log`)

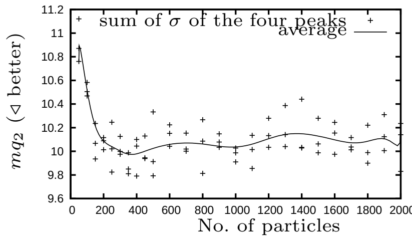
4 Experiments

4.1 Map quality measures

To check how the map quality measures behave under controlled settings, we run the following experiment: Using the identical log file (from the RoboCup Rescue final in China, 2008), we produced hundreds of maps but used different parameters in the map building process. For example, we iterated the number of particles that are used for the robot localization; for each particular number



(a) Map quality measure mq_1



(b) Map quality measure mq_2

Fig. 6. Map quality measure mq_1 and mq_2 versus number of particles (data set: `rescueServer_2008.07.20_13-49-13.log`)

we performed three complete mapping runs. From experience, we know that the more particles are used to represent the posterior of the robot pose, the more accurate the generated maps are. Typically, we used about 2,000 particles for our mapping. For the experiments, we iterated the particle count from 50 to 2,000. Fig. 6(a) shows the correlation between the number of particles and mq_1. Two conclusions can be drawn from the graph: 1. The gain in quality is large up to about 1,000 particles. 2. If the experiment is repeated, the resulting quality can vary significantly (e. g. see the three different values at particle count 1,400). So building maps concurrently definitively makes sense! Fig. 6(b) shows the correlation between the number of particles and mq_2. Here also up to a number of about 300 particles, the increased particle count yields a gain in quality. Then again – and here more significantly than with mq_1 – the resulting quality varies a lot, even with the same number of particles. This means: Even if you choose the same parameters, the outcome of the probabilistic algorithm can be very different. But now, using mq_1 and mq_2, we can detect the resulting quality.

4.2 Hyper Particle Filter

We tested the HPF using the log files from last years RoboCup Rescue autonomy final. Using mq_1, the HPF always picks the best map out of these 30 generated maps after each mapping step. This way, the probability of a highly accurate map was dramatically increased. Fig. 7 shows how the map quality increases with the number of particles, for one (bottom line) and for 30 concurrent Particle Filters (middle), and also the influence of the resampling (top line). Within the resampling step, SLAM particle filters with low quality maps are replaced by better particles, so that the average map quality for all particle counts can be improved further.

4.3 Loop closing

Using mq_1 the problem of loop closing can be solved. The loop closing happens when the robot drives around a block and returns to a spot that it has visited

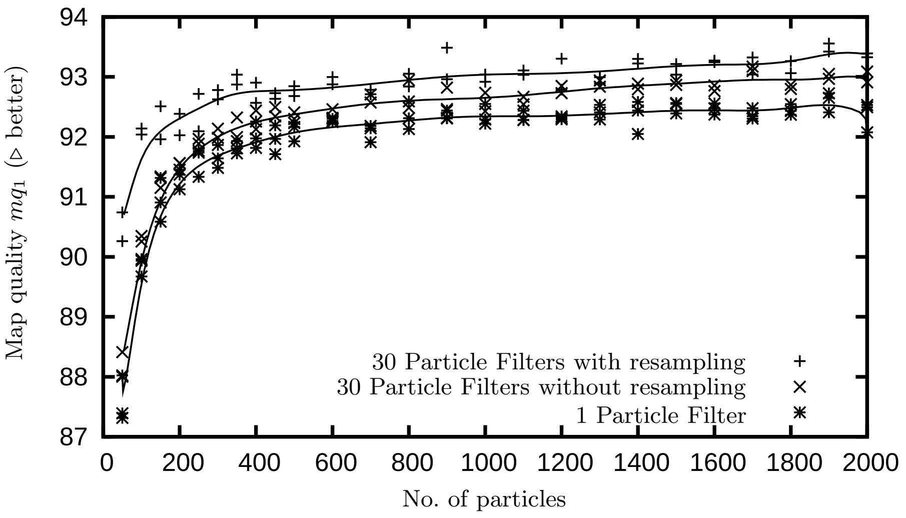


Fig. 7. Map quality measure mq_1 versus number of particles; 1 (lowest line) and 30 Particle Filters; without (middle) and with (top line) resampling (replacement of maps with low quality).

before using a different way. Accumulated errors often cause the map to break in these areas due to the chain error. For the experiments, we recorded data in a building in which a RoboCup Rescue arena is placed. We collected the odometry and laser range data while driving the robot around this arena. Immediately after the loop closing, the map was broken for a moment; this situation is illustrated in Fig. 8(b). Using mq_1, the broken map was detected. Then, the algorithm switch to another map (with a higher mq_1 value). This map is shown in Fig. 8(c). Note that in this map the walls fit exactly. The weights of the involved particles (only 3 out of the 30) is shown in Fig. 4.3. The quality of all maps drop after the loop is closed, but some particles can cope the situation better than others. So the HPF switches to a particle with a clearer and more consistent map.

The performance of the loop closing depends on the number of particles in the HPF: The more particles (and maps) are maintained concurrently in the HPF, the more likely a loop can be closed successfully.

5 Conclusion

In this work, we presented the concept of a Hyper Particle Filter for robust online mapping: A Particle Filter that is composed of a number of conventional SLAM Particle Filters. Two new map quality measures for the evaluation can be used for the measurement function of the Hyper Particle Filter: mq_1 determines the contrast of the map, while mq_2 calculates the distribution of the orientation of wall pixels calculated by the Sobel operator. While mq_1 is model free, mq_2

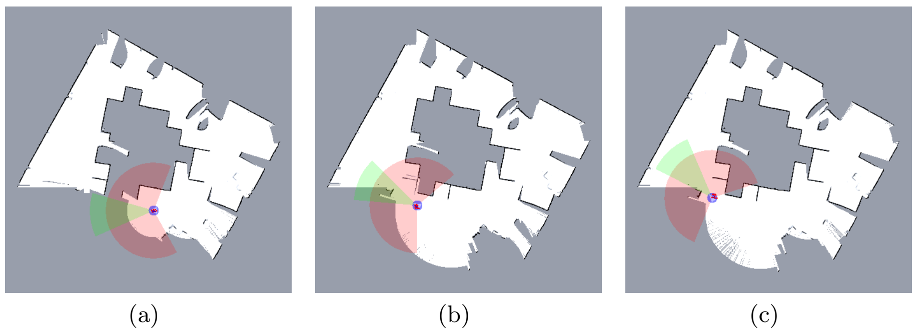


Fig. 8. Successful loop closing using a Hyper Particle Filter (HPF) with 30 particles. (a) Situation right before the loop closing. (b) The robot scans areas that it has visited before; first errors in the map are visible. (c) The HPF closes the loop by switching to a consistent map.

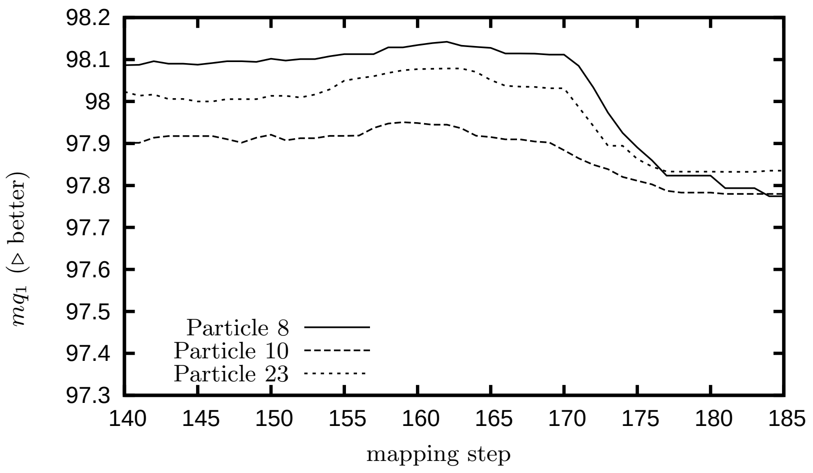


Fig. 9. Map quality measure mq_1 versus the mapping steps while the loop closing happens between steps 170 and 180: The quality of the map of particle 8 drops, but the map of particle 23 can cope with the loop closing.

relies on a orthogonal building structure assumption. Using these measures, we can detect broken maps and replace them by maps that are consistent.

We showed that the overall map quality is increased compared to the standard "single" SLAM Particle Filter by just using multiple Particle Filters concurrently. Using a simple resampling strategy, the overall quality could once again be increased. Because loop closings also influences the map quality, loop closing situations can implicitly be detected and a another map (that is consistent – even after the loop closing) is chosen automatically. We will use the new filter during RoboCup 2009 on our robot "Robbie 12" in the RoboCup Rescue league. In the test with last years log files, the quality of the maps generated by the HPF outperformed our last years (league's best) maps. With the data acquired in the larger structure, Robbie was able to close loops in the map. Due to a highly efficient implementation, the algorithm still runs online during the autonomous exploration.

6 Future work

So far the diffusion step of the Hyper Particle Filter is not implemented. We plan to modify parameters of the SLAM Particle Filters in this step: For example, the scan matcher that reduces the error of the odometry might be switched on or off. This way, the scan matcher is turned on automatically when it increases the accuracy of the mapping, but is switched off (and therefore saves computation time) when it is not needed. So the software can adapt itself depending on the current environment.

References

1. Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russel. Rao-blackwellised particle filtering for dynamic bayesian networks. 2000.
2. Hugh Durrant-Whyte. Localisation, mapping and the slam problem. Technical report, The University of Sydney, Summer School, 2002.
3. A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 6 1989.
4. J. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 318–325, Monterey, California, 1999.
5. D. Hähnel, W. Burgard, D. Fox, and Sebastian Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 1:206–211 vol.1, 10 2003.
6. Dirk Hähnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. A highly efficient fastslam algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.
7. M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.
8. Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.
9. Sebastian Thrun. Robotic mapping: A survey, 1 2002.
10. Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *ICRA*, pages 321–328. IEEE, 2000.