Proc. 3rd European Conference on Mobile Robots (ECMR) 2007, Freiburg, pp. 138-143, Albert-Ludwigs-Universität Freiburg - Universitätsverlag 2007

Memory-Efficient Gridmaps in Rao-Blackwellized Particle Filters for SLAM using Sonar Range Sensors

Christof Schröter Hans-Joachim Böhme Horst-Michael Gross*
Neuroinformatics and Cognitive Robotics Lab, Ilmenau Technical University, 98684 Ilmenau, Germany

***Abstract*— Simultaneous Localization And Mapping (SLAM) has been an important field of research in the robotics community in recent years. A successful class of SLAM algorithms are Rao-Blackwellized Particle Filters (RBPF), where the particles approximate the pose belief distribution, while each particle contains a separate map. So far, RBPF with landmark based environment representations as well as gridmaps have been shown to work. Existing gridmap approaches typically used laser range scanners, because the high accuracy of that sensor keeps the state uncertainty low and allows for efficient solutions. In this paper, we present a combination of our previous work on map-matching with RBPF, which enable us to solve the SLAM problem also with low-resolution sonar range sensors. Furthermore, we introduce a simple and fast but very efficient shared representation of gridmaps which reduces the memory cost overhead caused by inherent redundancy between the particles. An experimental comparison to a plain gridmap implementation shows the effective limitation of memory in loop-wise exploration.**

***Index Terms*— SLAM, Rao-Blackwellized Particle Filter, Occupancy Map**

I. Introduction and Related Work

In order to navigate autonomously, a basic requirement for a mobile robot is the ability to build a map of the environment. Because mapping depends on a good estimate of the robot's pose w.r.t. the environment, while localization needs a consistent map, the localization and mapping problems are coupled in applications where an unknown area has to be explored without an external position reference like GPS. The term Simultaneous Localization And Mapping (SLAM) has been coined for this problem [1]. The mutual dependency between pose and map estimates requires to model the state of the robot in a high-dimensional space, consisting of a combination of the pose and map state. SLAM can be seen as a generalization of the map building problem, as it describes the objective of aquiring a map of the environment without assuming any additional position information apart from those that can be derived from the mapping process itself. Therefore, in recent literature, SLAM is sometimes also referred to just as mapping.

There are two main criteria that can be used to categorize existing SLAM techniques: the kind of model used to describe the robot and environment state and the algorithm that is utilized to estimate the state belief.

*This work is partially supported by TAB Grant #2006FE0154 to H.-M. Gross

In many SLAM approaches, the map representation is assumed to be a vector of point-like feature positions [2]. These features correspond to landmarks in the environment which can be extracted by adequate feature-extraction algorithms. The attractiveness of feature/landmark-based representations for SLAM lies in their compactness. However, they rely on *a priori* knowledge about the structure of the environment to identify and distinguish potential landmarks. Furthermore, a data association problem arises from the need to robustly recognize landmarks not only in local vicinities, but also when returning to a position from an extended round-trip.

In contrast to landmark representations, gridmaps [3] do not make assumptions about specific features to be observable in the environment. They can represent arbitrary environment structures with nearly unlimited detail. However, they require a large amount of memory. In particular, the memory cost grows with the desired level of detail. Usually, 2D grids are used, but 3D grid approaches also exist. However, 2D grids are sufficient for most environments and applications of a wheel-driven mobile robot, so we will only regard 2D mapping here.

The estimation algorithms can be roughly distinguished in two main classes: Kalman filters and derivations thereof, and (Rao-Blackwellized) particle filters. The Kalman filter and its non-linear derivation Extended Kalman filter (EKF) as well as similar approaches like the Information filter [4] are best suited for landmark representations. They are able to estimate the full posterior distribution over the pose and map state. However they assume Gaussian distributions in the motion and observation model and become unstable if these assumptions are not met. More importantly, calculation of the full covariance matrix can become very expensive for large environments.

An effective means of handling the high-dimensionality in the SLAM problem has been introduced in the form of the Rao-Blackwellized Particle Filter (RBPF): in this approach the state space is partitioned into the pose and map state. A particle filter approximates the pose belief distribution of the robot, while each particle contains a map which represents the model of the environment, assuming the pose estimation of that specific particle (and its history, which is the estimation of the entire robot path) to be correct.

The RBPF was first proposed as a solution for the SLAM problem by Murphy et. al. [5], [6]. Montemerlo et. al. [7] used the RBPF for an efficient solution of SLAM on landmark maps, utilizing a laser range scanner for landmark identification.

The RBPF also provides a framework for using gridmaps in SLAM. Since the memory cost of gridmaps is comparatively high and each particle carries its own map, the number of particles is quite limited when using a plain version of this kind of model. Successful implementations mainly evolved from laser scan matching approaches: here, due to the use of scan registration techniques, the uncertainty in the robot pose can be kept very low during map aquisition. The small remaining position variance can be handled efficiently by a RBPF using a low number of particles. The high accuracy of the laser range scanner also allows to very accurately identify the correct (best) pose hypothesis when encountering differences between expected and perceived environment at a loop closing point.

Hähnel [8] was the first who adapted the FastSLAM algorithm of Montemerlo to gridmaps, introducing the inclusion of scan matching to decrease the uncertainty in the robot motion model. Grisetti [9] further improved that approach by adding the inverse sampling strategy also known from Mixture MCL [10], considerably decreasing the number of particles needed to robustly track and converge the map estimation. Eliazar and Parr [11], [12] chose a different way by identifying the redundancy in the separate particle maps that arises from resampling in the particle filter, which effectively means generating copies of good particles and their maps. By introducing an original storing scheme for the particle gridmaps which preserves the inheritance relation between all particles, they were able to reduce the memory cost significantly.

Our intention here has been to build maps with a robot equipped with odometry and a sonar range sensor array only. We therefore abandon any assumption of high-resolution range data like from a laser range scanner. Against this background, we can not use a scan registration or similar method to correct the odometry readings of the robot. Instead, we use the raw odometry data for forward propagation of the pose belief when the robot is moving through unexplored space. Obviously, the pose uncertainty which in turn determines the number of particles needed, depends on the accuracy of the robot odometry. We present experimental results with our new robot platform SCITOS A5 (Fig. 1), which features a quite accurate odometry.

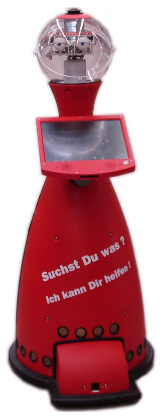

Fig. 1: Robot platform SCITOS A5 by MetraLabs GmbH

The main difference between the approach presented here and the previous solutions using laser range scanners lies in the comparison between actual environment observations and expectations from the map. While the laser approaches used a single scan to compare it against the map, we experienced that this does not lead to robust results with sonar, due to the lower resolution and significantly higher variances in the measurements. We already presented a solution for this problem in our previous work on map building [13], where we proposed a method of map matching for determining the best-matching pose given a sequence of sonar range readings (and relative positions from odometry) and a map of the environment.

The abandonment of scan matching in the forward propagation increases the number of particles needed to ensure robust loop closing, and therefore the memory cost for storing the particle maps. However, memory can be saved by exploiting redundancy: In the resampling process (most obviously at a loop closing point), irrelevant particles are deleted while good particles are cloned. This results in an identical copy of the particle map. Afterwards, the copies of the particle diverge and add different modifications to their respective maps. In a large environment, the differences often only affect a small part of the map though, which means that large areas of the maps belonging to different particles remain identical. By representing a map as an array of small grid patches instead of one large field, we are able to store each unique area patch only once independently of the number of copies, significantly saving memory.

By combining map matching and shared gridmap implementation with the RBPF framework, we are able to robustly build consistent environment maps from odometry and sonar without the need for a laser range scanner or similar high resolution data. The rest of the paper is organized as follows: We give a short introduction to the RBPF approach for SLAM in the next section. Section III will explain the specific details of our Sonar-SLAM implementation, while section IV deals with the shared gridmap representation. Experiments with real robot data are presented and discussed in section V, section VI closes with a short summary and outlook.

II. Rao-Blackwellized Particle Filter for SLAM

As already described before, the complexity of the SLAM problem arises from the very high-dimensional state space, consisting of the variables describing the robot pose and the variables describing the environment state. In the case of gridmaps, the map alone usually contains a few thousands up to several million cells, each of which corresponding to a state variable. Obviously, a full posterior over the state is extremely costly to estimate. The idea of the RBPF in application to SLAM is to use a particle filter to estimate the robot trajectory distribution $p(x_{1:t}|z_{1:t}, u_{0:t})$ given the sequence of odometry measurements $u_{0:t}$ and environment observations $z_{1:t}$. This trajectory estimate is then used to estimate the desired distribution over map and trajectory:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) = p(m|x_{1:t}, z_{1:t})p(x_{1:t}|z_{1:t}, u_{0:t}) \quad (1)$$

The particle filter works analogous to Monte-Carlo-Localization [16], except that instead of one given map each particle contains a separate map. To calculate the importance weights for $p(x_{1:t})$, each particle uses its own map. The map, in return, is built from the estimated trajectory of that corresponding particle. The effect is that a number of hypothesis maps are built, each corresponding to a possible trajectory. Importance weighting is performed with the weight for particle i following

$$w^{(i)} \simeq \frac{p(x_t^{(i)}|z_{1:t}, u_{0:t})}{\pi(x_t^{(i)}|z_{1:t}, u_{0:t})} \quad (2)$$

Here, $\pi(x_t^{(i)})$ denotes the proposal distribution. Typically, the motion model is used to generate the proposal distribution from the last particle generation (again, in analogy to localization), in which case the weight formula simplifies to

$$w^i \simeq p(z_t|x_t^{(i)}, m^{(i)}) \quad (3)$$

By repeatedly calculating importance weights followed by resampling to adapt the particle distribution to the estimated distribution, particles are preferred whose maps match new observations best, therefore the most likely map is selected. Subsequently, we present a specific way of calculating the particle weights for unreliable sensors where scan matching is not applicable.

III. Sonar Grid SLAM

The base of our Sonar SLAM approach is a particle filter, where each particle contains a pose (x,y, heading ϕ) estimate as well as a map estimate. Without loss of generality we can assume the robot to start mapping at position (0,0,0). However, it is also possible to initialize the particle filter at any other pose or even with any pose distribution, e.g. to align with a previously learned part of the map. While the robot moves, the particles move as well, according to the odometry readings and the probabilistic odometry motion model, which describes the uncertainty in the actual robot motion. Due to this uncertainty, the motion model contains a stochastic component, which effects in the particles spreading out and generating slightly different trajectories. Basically, we use the standard motion model which also can be found in [8]. Additionally, during motion the robot observes the environment by means of sonar range sensors. A map update is triggered frequently (approx. every 0.2m). In that map update, each particle adds the new environment observation to its own map, at its own estimated current position, using the standard Bayesian occupancy update method for gridmaps [3]. Since the position estimates of the particles are slightly different, the maps differ as well (Fig. 2).

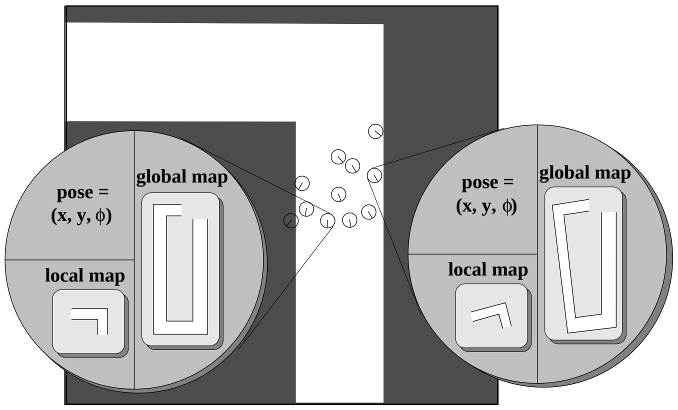

Fig. 2. Data representation overview: The particles model the distribution of the robot pose belief. Each particle contains a full map of the environment, which is a combination of the full particle trajectory and the sonar range measurements. Furthermore, each particle contains a local map, which only contains the most recent measurements and depends on the particle's current pose belief. The situation shown is shortly before a loop closing. Apparently, the left particle is a better approximation of the true pose than the right particle.

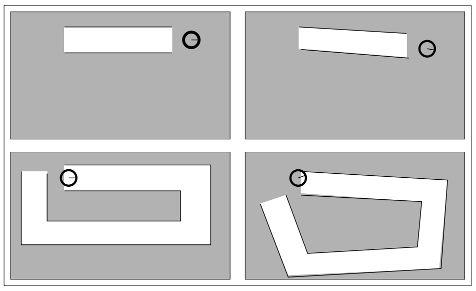

Fig. 3. Comparison of good (left) and bad (right) pose estimation. Each of the images shows the estimated pose and map of one of 2 particles, assuming the left one to be correct. The upper row shows the 2 particles at an early time. The pose estimation of the right particle deviates from the true state. However, since both particles do not know their current environment, they get assigned the same weight. In the lower row, the loop closing situation is shown for the same 2 particles. Only now the correctness of the pose estimation can be evaluated by comparing the actual observation to the expected ones.

It should be noted that due to the low resolution, the relatively high variance, and particularly the low reliable maximum range of a sonar sensor in comparison to a laser scanner more updates are needed for each cell to achieve a reliable occupancy estimate. While a laser range scanner can build a good local map from a single range scan, several measurements from different positions are needed with sonar sensors. An immediate effect is that when exploring unknown space, the area in front of the robot is not approximated well and the robot can only "look ahead" for a very limited distance. This is usually not a severe problem for map building since it can be compensated easily by moving accordingly, but it becomes important in the particle weight calculation: here we need a weighting function for the particles to select good estimates and reject those particles that represent an unlikely state. To this purpose, the internal map representations of the particles are continuously compared to the actual observations.

However, since the map in a particular position is unreliable until the robot has actually moved beyond it, we do not regard the most recent map updates for the weight calculation and rather consider the map to be undefined at the current robot position, unless the robot already passed there earlier. In the actual implementation, this is realized by delaying updates of the global map until the robot has moved on for several meters, using a queue for the range measurements and a position queue in each particle. In effect, from the start of mapping, as long as the robot moves forward, for all the particles the map is unknown at the current position and therefore the expected observations of all particles are the same. Only when the robot returns to a known area (more precisely, when a particle believes to return to a known area), a map approximation for the current position exists and different particle weights will occur (Fig. 3).

For the actual weight calculation, initially we tried generating an expected range scan from the map and using a model of the observation probability $p(d_{observed}|d_{expected})$ built from sonar range data beforehand. We had achieved good results with this method in a localization algorithm where a map was given already [17]. However, it did not lead to success in SLAM, in particular because here we need to handle situations where the particle does not have any map knowledge at the current believed position. Problems mostly arise when the robot returns to a position it has already visited before and some of the particles are "aware" of the loop closing while others are not: Those particles that represent a good position belief will have a certain expectation for the range measurement, while many of the bad particles will still believe to be in unknown space and not be able to generate a range expectation. A consistent weight calculation apparently was not possible for such situations.

We already experimented with a different way of comparing expectation and observation from sonar range sensors in a previous work on mapping [13]. There, we proposed a map matching approach: a local map is built from only the most recent sonar measurements and the resulting local map is matched against the global map to find the most likely position w.r.t. that global map. A similar technique was utilized in [14]. In contrast to the particle filter we present here, only one pose hypothesis was used in those approaches, which was iteratively corrected by searching the local maximum of the match function. In that maximum search, ambiguities are likely to arise in straight corridors, where a position can be chosen arbitrarily along the direction of the hallway and small uncertainties often added up, leading to large position errors eventually. However, with the SLAM approach presented here this is not a problem because we do not need to explicitely search a maximum match position: the position hypotheses are represented by the particle filter and the match value is only used to give a weight for each particle effectively modeling an entire distribution of the match function.

There are two more advantages of map matching instead of directly using the probability distribution $p(z|x,m)$. First, it is a tedious task to determine the full distribution of measurements conditioned on real distance $p(z|d)$ for a given sensor, while for mapping it is usually sufficient to know just the expected true distance for a certain measurement - which should be equal for a suitable sensor. Furthermore, a map is also capable of incorporating sensor observations for which such a probability distribution can not be defined easily, e.g. the occupancy map could be generated by a visual scene reconstruction algorithm like the one presented in [15]. We are currently working on implementing "Visual SLAM" on this base.

In order to be able to use map matching, each particle must not only know its global map, but also a local map. Since we already exclude the most recent range measurements from the global map, we can use those measurements for the local map. That way, global and local map are built from different data and we avoid comparing certain measurements against themselves. The local map can either be rebuilt from the pose and scan queues for each weight calculation or be persistent in the particle by just adding every new scan and forgetting old scans. Making the local map persistent is more efficient but less flexible. In both cases, the local map depends on the believed recent trajectory of the particle and can therefore be considered as an integral part of the particle (Fig. 2).

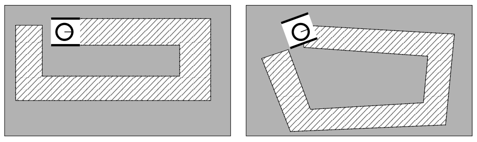

Fig. 4. Map matching: For the left particle, representing a correct position belief, the local map (clean white) is aligned to the global map (hatched) very well, while for the right particle, which does not contain a position belief consistent with the environment, the local map conflicts with the global map (many of the occupied wall cells in the local map correspond to free cells in the global map). This situation would result in a higher weight for the left particle, supporting the position hypothesis which generates a consistent map.

The calculation of the match value between the local and global map is quite simple: For each occupied cell in the local map the occupancy value of the corresponding cell in the global map is tested. If the global map cell also is occupied, that cell contributes with a value of $+1$. If the global map cell is free, it contributes with a value of -1. Only cells with a value above or below a threshold (occupied/free) contribute to the match value. That way, the match value is positive if local and global map are very similar, and it is negative if many objects exist in the local map where there is free space in the global map. To obtain the actual particle weight $match^{(i)}$, an exponential function is applied as follows:

$$w^{(i)} = e^{\frac{match^{(i)}}{f}} \tag{4}$$

with f being a free parameter to influence the spread in the particle weights and therefore the speed of convergence.

IV. Shared Gridmaps

A major problem with using gridmaps in RBPF is the memory cost: In a naive implementation, the number of cell values to be stored would be the product of grid size and particle number. However, the maps of the individual particles are not completely independent: In the resampling as part of the observation update, particles with low weights are deleted

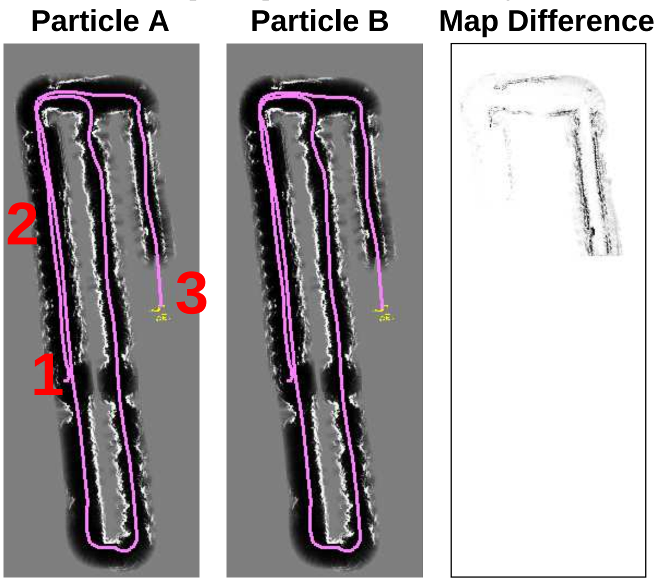

Fig. 5. The robot started at position 1, closed the loop and moved on to position 3. Particle B was generated as a copy of A during resampling approximately at position 2. Therefore, the major part of the map is identical between particles A and B. The right image highlights the map differences.

and replaced by copies of particles with higher weights. This results in multiple identical copies of the same map. Afterwards, each of the particles will modify its respective map differently, according to the path assumed through the probabilistic motion model: The copies will not remain identical, but it is important to notice the changes often only affect a small area of the already aquired map (see Fig. 6). The idea to save wasting memory for redundant information therefore is to split up the map into smaller patches and share those patches across the particles. When a particle A is cloned, each ”copy” of a map patch belonging to the clone particle B is just a reference to the original patch. Only when either A or B modify a map patch later, a real copy is created in the local memory of the respective particle. If particle A or B is copied again without having modified a certain patch yet, the new particle C will create a third reference to that same patch, etc. Each patch will continue to exist until *all* references to it have been deleted.

The effect of this representation is that the memory cost is not determined by the map area, but by the size of path loops. As long as a loop is not closed yet, particles are diverging and many path hypotheses are maintained. When the loop gets closed, only the best particles survive, and new particles are generated as copies of those few best fitting hypotheses. While a loop is open, each particle holds an own independent map of that loop, but when it is closed, only few unique maps of that specific loop (the best fitting ones) continue to exist. Therefore, the ”residual” memory cost is determined by the entire map area (the sum of all loops) and nearly independent of the particle number, while the peak memory cost is determined by particle number and maximum length of a single loop.

Obviously, the presented approach is somewhat reminiscent of DP-SLAM by Eliazar [11], [12]. However, in contrast to their method we do not explicitely maintain particle ancestry and do not track modifying access for each particle at cell detail, but for constant size blocks (referred to as map patches). This results in a significantly simpler implementation. Although the handling of redundancy is coarser and, therefore, computational and memory cost reduction will be slightly

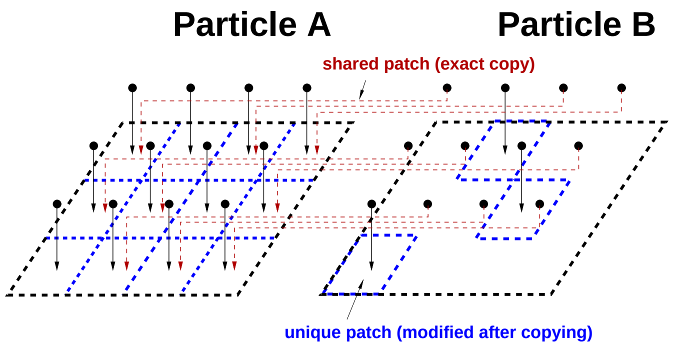
Particle A
Particle B
shared patch (exact copy)
unique patch (modified after copying)

Fig. 6. Shared map representation: Particle A contains a map which consists of a number of separate patches, where each patch is a gridmap of an area of about 10 * 10 m. Particle B is created as a copy of A: The map of B consists of references to the patches in A. When A or B modify a certain patch, it creates a real copy first, so Particle A and B have a separate instance of that patch. In the situation shown, 3 patches have been modified after B was copied from A. Therefore, those 3 patches exist separately in each particle, while all other patches exist just once and are shared between the particles.

smaller, the overall goal of limiting peak memory is reached with lower effort.

We recently learned that our technique of sharing map patches bears some similarities to a method presented by Grisetti et.al. [19]. There, they partioned the map into patches according to the robot path, while in our implementation sharing is completely handled in the map itself and transparent to the particles, which is favorable from an implementation point of view.

Furthermore, one might argue that the convergence to a small number of valid hypotheses at the loop closing could be handled by actually reducing the particle number, which in turn would automatically minimize the memory cost at that time. However, this would not solve the problem actually: With growing uncertainty in the next open loop, more particles would be needed, which again would be generated by cloning existing particles and their maps in particular. Therefore this would not affect peak memory requirements. Nevertheless, an adaptive particle number is expected to reduce computational cost and is going to be implemented in the near future.

V. Experiments

To test our approach, we built maps of a home store which is the regular test environment for our navigation algorithms [18]. This environment is very well suited for our proposed SLAM approach as it essentially consists of a large number of small circles of hallways (50 to 100 m loop length) (Fig. 7 top row). Only the robot odometry and sonar range sensors were used in those experiments.

A map update was not done with every observation, but in intervals. Whenever the robot has moved on for 0.2 meters, first the new particle position is sampled from the odometry motion model, then the particle's map is updated with the observation at that believed position. The images show the trajectories of all particles. An importance weight calculation and resampling of the particles was done in intervals of 1 meter. The parameter f from eq. 4 was set to the value 100, therefore $w^{(i)} = exp(0.01 \cdot match^{(i)})$.

VI. Summary & Outlook

We presented an implementation of RBPF with gridmaps which is able to solve the SLAM problem with low-resolution sensors such as sonar range finders. Furthermore, we introduced a shared map representation for particle filters which effectively makes the maximum memory cost dependent on the loop size instead of the overall map size. Experiments show that our approach is well suited for large-scale environments ($100*100$ m^2) consisting of many loops with a limited length. Implementation on a semi-automatic exploration method for integration in SLAM is currently in progress and will be presented in the near future.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, The MIT Press, 2005.

[2] R. Smith, M. Self, and P. Cheeseman, “A stochastic map for uncertain spatial relationships,” in *4th International Symposium on Robotic Research*. 1987, MIT Press.

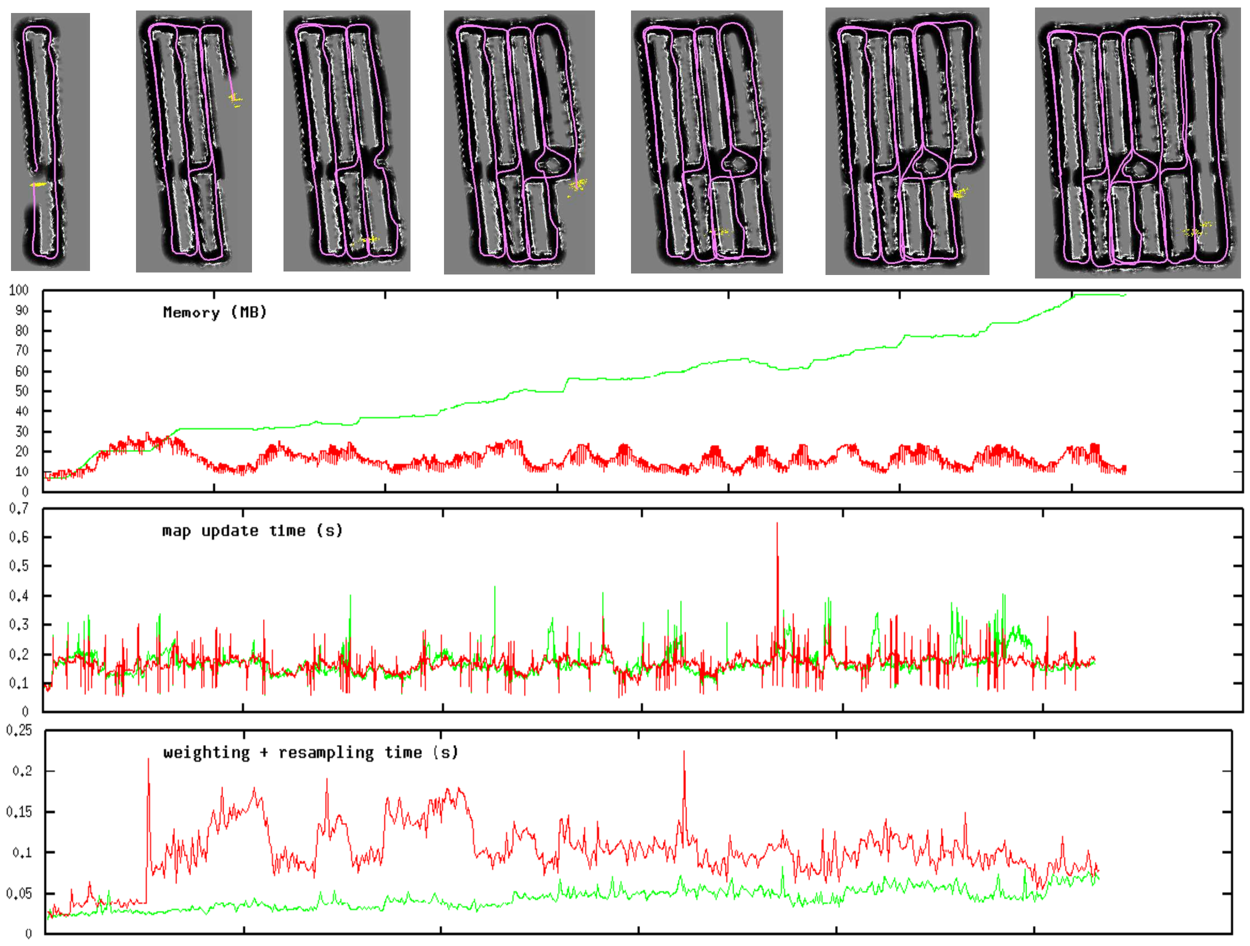

Fig. 7. Row 1: Several steps of mapping (500 particles): Yellow dots are the particle positions, path (magenta line) and map for one particle are shown. Row 2: Map memory cost for plain gridmaps (green) and shared maps (red, see section IV). It is clearly visible that the memory for plain maps is growing monotonously with the mapped area, while for the shared maps the cost collapses with each loop closure (however, it will also grow unbounded).
Row 3+4: Map update time and weighting/resampling time for plain gridmaps (green) and shared maps (red). The average time for map updates is nearly identical. Weighting takes longer for shared maps as the access of cell values is a bit more costly. However, weight updates occur significantly less often than map updates, therefore in the overall process time the effect is nonsignificant.

[3] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI Magazine*, vol. 9, no. 2, pp. 61–77, 1988.
[4] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and Ng. A.Y., "Simultaneous mapping and localization with sparse extended information filters," in *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics*, 2002.
[5] K. P. Murphy, "Bayesian map learning in dynamic environments," in *Proc. NIPS99*, 1999, pp. 1015–1021.
[6] A. Doucet, N. de Freitas, K. Murphy, and S. Russel, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Sequential Monte Carlo Methods in Practice*, N. de Freitas A. Doucet and N.J. Gordon, Eds. Springer-Verlag, 2001.
[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002.
[8] D. Haehnel, W. Burgard, D. Fox, and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Proc. IROS-2003*, pp. 206–211.
[9] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in *Proc. ICRA-2005*, pp. 2443 – 2448.
[10] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo Localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.
[11] A. I. Eliazar and R. Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *Proc. IJCAI-2003*, pp. 1135 – 1141.
[12] A. I. Eliazar and R. Parr, "DP-SLAM 2.0," in *Proc. ICRA-2004*, pp. 1314 – 1320.
[13] C. Schroeter, H.-J. Boehme, and H.-M. Gross, "Robust map building for an autonomous robot using low-cost sensors," in *Proc. SMC-2004*, pp. 5398 – 5403.
[14] A.C. Schultz, W. Adams, and B. Yamauchi, "Exploration, Localization, Navigaation and Planning with a Common Representation," in *Autonomous Robots*, vol. 6, no. 3, pp. 293 – 308, 1999.
[15] E. Einhorn, C. Schroeter, H.-J. Boehme, and H.-M. Gross, "A Hybrid Kalman Filter Based Algorithm for real-Time Visual Obstacle Detection," in *Proc. ECMR 2007, this volume*.
[16] D. Fox, W. Burgard, F. Dellaert and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots" in *Proc. AAAI Natl. Conf. on Artifical Intelligence*, 1999, pp. 5398 – 5403.
[17] C. Schroeter, A. Koenig, H.-J. Boehme, and H.-M. Gross, "Multi-sensor Monte-Carlo-Localization combining omnivision and sonar range sensors," in *Proc. ECMR-2005*, pp. 164–169.
[18] H.-M. Gross, A. Koenig, H.-J. Boehme, and C. Schroeter, "Vision-based Monte Carlo self-localization for a mobile service robot acting as shopping assistant in a home store.," in *Proc. IROS-2002*, pp. 265–262.
[19] G. Grisetti, G. Tipaldi, C. Stachniss, W. Burgard and Daniele Nardi, "Fast and Accurate SLAM with Rao-Blackwellized Particle Filters," in *Journal of Robotics and Autonomous Systems*, vol.55, no. 1, pp. 30 – 38, 2007.