2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Fast Monte Carlo Localization Using Spatial Density Information

Renan Maffei[1] and Vitor A. M. Jorge[1] and Vitor F. Rey[1] and Mariana Kolberg[1] and Edson Prestes[1]

Abstract— **Estimating the robot localization is a fundamental requirement for applications in robotics. For many years, Monte Carlo Localization (MCL) has been one of the most popular approaches to solve the global localization when using range finders, like sonars or lasers. It generally weights the estimates about the robot state by comparing raw sensor readings with simulated readings computed for each estimate. In this paper, we propose an observation model for localization that associates a kernel density estimate (KDE) to each point in the space. This single-valued density measure is independent of orientation, what allows an efficient pre-caching step, substantially boosting the computation time of the process. Using the gradient of the densities field, our strategy is able to estimate orientation information that helps to restrict the localization search space. Additionally, we can combine densities obtained by kernels of different sizes and profiles to improve the quality of the acquired information. We show through experiments in comparison with traditional approaches that our method is efficient, even working with large sets of particles, and effective.**

I. Introduction

Mobile robot localization is one major area of study in robotics [1]. A robot cannot properly interact with the environment or execute relevant tasks without knowing its position in the world. While simultaneous localization and mapping (SLAM) and integrated exploration techniques are able to place the robot inside a, typically, on-line constructed map [2], [3], such methods are often unnecessary in previously mapped areas and man-made structures. Robots can access adapted versions of such maps prepared for mobile robot localization.

Localization can be seen as tracking a robot position from a previously known position or globally localizing it in a map [4]. In order to solve the localization problem most approaches model the robot position as a probability distribution, such as Gaussians [5], histograms [6] or particle filters [7]. In these cases, the Bayesian approach is adopted, where the previous robot state is assumed to be sufficient to infer the next one. At each step, a prediction about the robot state is made respecting the robot motion model. Then, readings from sensors are used to correct the estimate using a sensor measurement model.

The modeling using unimodal Gaussian strategies, either to track single [5] or multiple hypotheses [8], usually works well with the assumption of small uncertainty. However they require mechanisms to extract salient features in the environment, such as geometric beacons [5], [8]. In contrast, non parametric approaches, such as histogram and particle filters, can work direct with raw sensor readings. Histogram filters precompute the distance from walls in every cell of a grid representing the map, from a discrete set of orientations. They can obtain high accuracy in the localization depending on the coarseness of the grid, nonetheless they may require large amounts of memory to store all available robot states in the map, which may as well imply in prohibitive computation time. Particle filters, as defined in the Monte Carlo Localization [7], solve the localization problem spreading particles (samples) throughout the map. Its cost is linear in the number of particles, but it can become expensive as the number of samples increases – e.g., in large maps. Still, particle filters are highly popular to solve the localization problem due to its underlying simplicity and robustness [4].

The efficiency of such localization strategies relies on the way sensor observations are modeled, among other things. In turn, the measurement model of a single reading of a range finder involves a probability distribution – or a combination of probability distributions – considering parameters such as the measured distance as well as sensor position and orientation. When using a sensor that performs multiple readings at a single time step – e.g. a laser range finder – the probabilities of all readings are combined to provide a single probability of a given position in the map. So, the observation probability must be computed for each hypothesis of the robot pose. Thus, it may become costly when the number of particles is large. Some alternatives to speed-up the process are the subsampling of the measurements and the pre-computation of the ray-casting in each discrete combination of position/orientation available in the map. However, depending on the scale of the map being processed, these modifications imply in reduction of precision and a substantial increase in memory requirements [4], [6].

In this paper we propose an efficient observation model focused on global localization and robot kidnapping problems. The algorithm basically computes and compares the discrete kernel density estimate (KDE) in the surroundings of the robot with those in the surroundings of the free cells in the map. The KDE is extracted considering the density of free space cells, since they are more abundant and less noisy than obstacles cells. Different kernel profiles and sizes can be used to detect different characteristics of the environment, such as proximity of walls, bifurcations, etc. The KDEs of each position in free space are precomputed and used to match the observations made by the robot in a given instant in time. Furthermore, it is orientation independent and has a low cost both in terms of processing time and memory requirements. Our method is particularly efficient to reach the pose tracking stage, due to the large number of samples it can handle in a very small period of time.

Institute of Informatics, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil rqmaffei,vamjorge, vfrey, mariana.kolberg, prestes@inf.ufrgs.br

This paper is presented as follows. Section II presents the related work. Section III introduces the proposed method. Section IV presents experiments and results supporting our claim, while Section V discusses the implications of our strategy, both in terms of accuracy and efficiency, and future directions.

II. Related Work

Observation models are used to evaluate how close an observation hypothesis is to the true robot observation, given the environment map. That is, they are be used to evaluate the quality of given hypotheses.

Early works in localization relied mostly on feature-based observation models, for instance extracting geometric beacons, such as walls and corners, from sonar data [5] [8]. Nowadays, feature-based methods are popular due to the use of cameras, making the definition of such models directly linked with the area of computer vision.

Nevertheless, range finders have been used in robotics for a long time and are still broadly in use. Fox et al. [9] describe a beam model considering noise in the measurement, errors due to unexpected objects and to failures to detect objects. The main problem of such approach is the ray-casting computation. Performing pre-caching reduces substantially the time cost, while increasing the space cost. Is important to note that the pre-caching occurs in 3D (x, y and θ), which may require significant growth in memory, depending on the chosen grid resolution [4]. Thrun et al. [10] propose the likelihood field model, which only evaluates the proximity to obstacles of sensor beams endpoints. Their method obtains fast smooth results, but since it only checks the endpoints of a reading, it allows the readings to pass through obstacles.

Another way to model sensor observations is with correlation measures. The idea is to build a local map surrounding the robot and compare it with the surroundings of the robot pose estimates. Duckett and Nehmzow [11] devise a technique which stores a pair of histograms over a discrete grid where each histogram bin counts the number of free cells as well as obstacles in horizontal and vertical directions in a square of 64x64 cells. A compass is used to rotate the orientation of the robot until it matches the map "north" to perform the measurement. Matching is performed convolving the two histograms obtained by the robot with every pair of histograms previously calculated and stored in each cell of the map, which provides a probability used to localize the robot in the environment.

Zhang et al. [12] define the concept of Similar Energy Region (SER). SER is a value associated to each position in the free-space, corresponding to the sum of the lengths of all readings made by the robot at such position. Since they use a robot able to get readings in $360°$, their measure of energy is virtually independent of orientation (this is only true under the assumption of a perfectly circular robot, with uniformly distributed sensors). Nonetheless, this idea of associating a single value to each position in space and use it to localize the robot is what lies in the core of our approach, as we will present next.

III. Using local kernels to localize a robot

In this section we detail the use of local kernels and their corresponding density estimate applied to the global localization problem.

A. Overview

Our localization strategy assumes that the local map obtained by a discrete set of laser range finders and odometry readings (coupled with a valid motion model) is locally consistent. This assumption is supported by evidences found in previous works of simultaneous localization and mapping [2], [3]. In our experiments, we consider the map as a discrete regular grid. Each cell in the grid map contains one or more kernel density estimates which are precomputed for future use.

To obtain the current observed kernel density estimate, the observation model requires a local map surrounding the robot position. If a local map is available, we can determine the free-space cells within the kernel radius using a simple flood-fill algorithm. When the flood-fill reaches unknown cells inside the kernel radius, the density will not be computed, because the density value would be different from the one obtained with the fully known map. Therefore, we cannot obtain the kernel density estimate using single observation of a range finder with $180°$. Instead, we use standard mapping techniques such as Histogramic In-Motion Mapping (HIMM), Bayesian approaches or the Dempster-Scheffer model [13][1]. In this paper, we construct the local map using HIMM. It is fundamental to avoid explicit use of ray-casting, since it can leave holes in the local map, due to sampling problems – as previously said, this would prevent the computation of our observation model. As other particle filter localization strategies, ours also demands robot motion and a proper motion model.

We only keep cell information up to a certain maximum range, D. Once a cell leaves the local map, its information is erased to keep the consistency of the local grid map (see Fig. 1). It is critical to observe that the local map and the stored maps have the same wall thickness, otherwise problems related to differences between online and stored densities will occur.

B. Local Kernel density estimate

In ideal conditions, the kernel density estimate Ψ of free space surrounding the point $\mathbf{p}_0 = (x_0, y_0)$ can be computed through Eq. 1

$$\Psi(\mathbf{p}_0) = \int s(\mathbf{p}) K(||\mathbf{p} - \mathbf{p}_0||)\ d\mathbf{p}, \tag{1}$$

where $\mathbf{p}$ is a point in the map and $K(.)$ is a kernel profile, while

$$s(\mathbf{p}) = \begin{cases} 1 & \text{, if } \mathbf{p} \text{ is inside a free space cell} \\ 0 & \text{, otherwise} \end{cases} \tag{2}$$

[1]If the local map is not reliable, one can make use of an online SLAM of choice to provide additional robustness to the map – e.g. a Rao Blackellized Particle Filter with few particles.

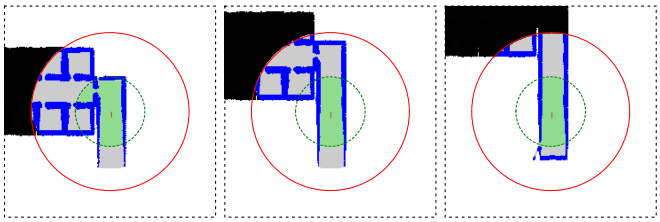


Fig. 1. Example of local window centered on the robot, in three different time steps. The red circle delimits the frontier of the local grid map ($D = 9m$), the green circle delimits the region under the kernel influence ($r = 4m$) and the light green region corresponds to the visited space covered by the kernel.

In the discrete case we use Eq. 3 to pre-compute the kernel density estimate for every cell in the grid map.

$$\Psi(\mathbf{p}_0) = \sum_{\mathbf{p}} s(\mathbf{p}) K(||\mathbf{p} - \mathbf{p}_0||) \tag{3}$$

Note that in practice, $\mathbf{p}$ is restricted to the area encompassed by the kernel radius. It is possible to use local map parameters – such as the HIMM count or the probability of a cell being occupied – to improve the results of Eq. 2 and 3.

Our approach was tested considering three different kernels profiles: uniform $\mathbf{K_u}(.)$, Gaussian $\mathbf{K_g}(.)$, and Inverted $\mathbf{K_i}(.)$. The first basically computes the area of the free-space region covered by the kernel, the second gives more weight to cells in the center of the kernel, while the third gives more weight to cells in the border of the kernel. Their responses are represented in

$$\mathbf{K_u}(d) = \begin{cases} a & \text{, if } d \leq r \\ 0 & \text{, otherwise} \end{cases} \tag{4}$$

$$\mathbf{K_g}(d) = \begin{cases} \exp\left(\frac{d^2}{2r^2}\right) & \text{, if } d \leq r \\ 0 & \text{, otherwise} \end{cases} \tag{5}$$

$$\mathbf{K_i}(d) = \begin{cases} r\left(b + c\left(1 - \sqrt{1 - \frac{d^2}{r^2}}\right)\right) & \text{if } d \leq r \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where r is the radius of the kernel and a is the height of the uniform kernel – typically $a = 1/(\pi r^2)$. For all kernels, d is the distance from the current cell being measured (i.e. the center of the kernel). For $\mathbf{K_i}(.)$, b and c are respectively the height of the uniform circular kernel and the length of the semi-axis of the oblate ellipsoid, with $b \approx 0.871c$.

To illustrate the motivation behind $\mathbf{K_i}(.)$, consider a kernel with width r and two robots, $\mathbf{R}_1$ and $\mathbf{R}_2$, in a corridor with length $l \gg r$. R_1 is positioned in the center of the corridor while R_2 is next to the wall. The densities computed through Eq. 1 using $\mathbf{K_g}(.)$ or $\mathbf{K_u}(.)$ with radius r will result in different values for $\mathbf{R}_1$ and $\mathbf{R}_2$, i.e. $\Psi(\mathbf{R}_1) > \Psi(\mathbf{R}_2)$. This is true for any monotonic decreasing kernel. However, this is not true for monotonic increasing kernels, such as $\mathbf{K_i}(.)$. Thus, we choose the values of b and c such that $\Psi(\mathbf{R}_1) = \Psi(\mathbf{R}_2)$. The use of $\mathbf{K_i}(.)$ has three important implications: the weight of free cells next to walls receive a large weight; the densities computed next to walls will not be different from that in the center of the corridor; and the density computed as robot $\mathbf{R}_2$ moves towards the center of the corridor will vary differently than those of $\mathbf{K_g}(.)$ or $\mathbf{K_u}(.)$. Thus, different kernels can result in different density values for the same position in space. This is particularly useful for localization since the combination of different kernels, e.g. monotonic increasing and decreasing kernels, may improve the distinction between places.

When we apply $\Psi(.)$ over the entire map, we obtain a scalar field of density estimates of free space cells.[2] The density estimates, along with orientation estimates computed using densities, are used to compute the weight $w(\mathbf{p})$ of each particle.

$$w(\mathbf{p}) = w_\Psi(\mathbf{p}) \cdot w_\alpha(\mathbf{p}) \tag{7}$$

The first term of the expression is computed by comparing the density value from the current robot position, $\Psi(\mathbf{r})$, to the one associated to the current particle position, $\Psi(\mathbf{p})$, which was pre-calculated

$$w_\Psi(\mathbf{p}) = \begin{cases} 1 & \text{, if } ||\Psi(\mathbf{r}) - \Psi(\mathbf{p})|| \leq \varepsilon \\ 0.5 & \text{, otherwise} \end{cases} \tag{8}$$

It is used to reduce the importance of particles with density more different from the measured by the robot than a threshold ε. We do not set the importance weight of the failed particles as 0, because the density information is not as reliable as other approaches using laser observations directly. For this reason, we set the weight of particles which failed the test as half of those which passed.

From the scalar density field it is also possible to obtain orientation information using standard gradient extraction methods, which we use to compute $w_\alpha(.)$. In this paper we use the Sobel operator, but one can also pre-calculate approximate orientations with methods such as used by Liu et al. [14]. To estimate orientation information using densities, we compute the angle difference, α_r, between the robot and the gradient of the scalar field. Then, for every particle, we obtain the difference between the orientation of the particle and the pre-computed gradient, α_p, stored in the map. The angle tolerance filter is calculated checking if the difference between α_r and α_p is smaller than a threshold t.

$$w_\alpha(\mathbf{p}) = \begin{cases} 1 & \text{, if } ||\alpha_r - \alpha_p|| \leq t \\ 0.5 & \text{, otherwise} \end{cases} \tag{9}$$

The Eq. 9 gives less importance to particles with incorrect angle differences. Unfortunately, angle cuts have shown noisy behavior next to and at local maxima, local minima and walls. Therefore, we compute the weight only in regions where the gradient of the densities is stable. Taking such precautions, and setting the threshold t to considerably large differences, the results become useful.

Finally, we combine different kernel density estimates with different radius to improve global localization results. The

[2]While we could have extracted density information for obstacle cells, the results obtained in informal experiments were highly unstable due to noise and sampling problems.

advantage of our method is the computation time – one order of magnitude faster than traditional methods –, which is an important factor for robots with limited processing power. Thus, the combination of a small number of kernels implies small extra computation time. The weighting of a particle considering multiple kernels is just the product of the individual weights

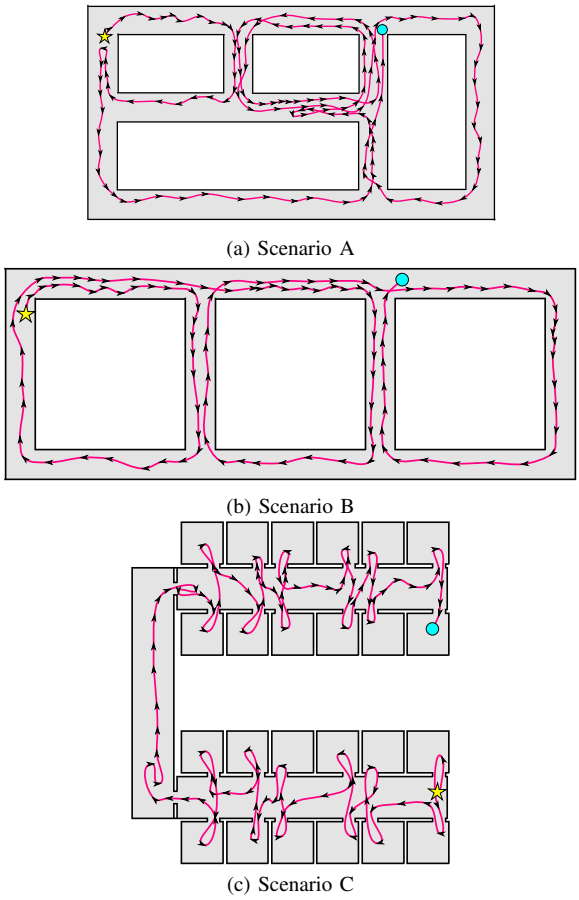
$$w(\mathbf{p}) = \prod_i w_i(\mathbf{p}). \tag{10}$$

IV. Experiments

Our experiments were conducted using a Pioneer 3DX equipped with a SICK LMS-200 laser range finder with $180°$ in simulation environments. The orientation error is set to $1°$, while the displacement error is set to 1%. In all experiments, a notebook with 8GB and an i7 processor was used to run the tests.

The proposed observation model was evaluated in three different scenarios (see Fig. 2). Scenario A contains four adjacent loops of different lengths ($44m$, $46m$, $58m$ and $78m$), while scenario B contains three adjacent loops of same length ($72m$). Scenario C contains several small rooms in two separate galleries, inside an area of approximately $25m \times 25m$.

All maps were discretized using a regular grid with $10\text{x}10cm^2$ cells. In all experiments, we set $D = 9m$, $\varepsilon = 0.2$ and $t = 45°$. Tests were divided in two subsections, the first considering only the proposed observation model, and the second showing a comparison with other strategies. For this experiment, we do not pre-compute the gradient angle for each of the stored density maps, but this could also be done, reducing the computation time even further.



(a) Scenario A

(b) Scenario B

(c) Scenario C

Fig. 2. Scenarios used in the experiments, showing the initial robot position (yellow star), final robot position (blue circle), robot path (pink) and path direction (black arrows).

A. Evaluating the Local Density Estimate Observation Model

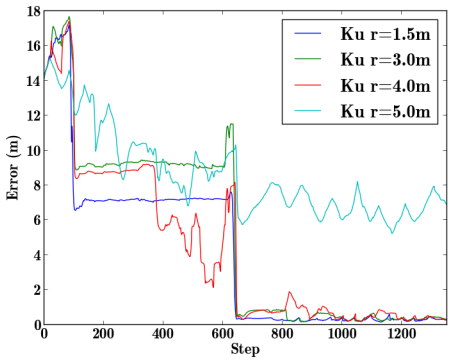
In this section, we present an evaluation of the different kernels regarding localization capabilities, as well as computation times. Figs. 3(a)-(c) show the mean localization errors for $\mathbf{K_u}$, $\mathbf{K_g}$, and $\mathbf{K_i}$ considering kernels with radius of $1.5m$, $3.0m$, $4.0m$, and $5.0m$. Fig. 3(d) presents three approaches, each one combining kernels of same profile and different radius ($1.5m$, $3.0m$ and $5.0m$). We performed 15 tests for each configuration, setting the number of particles to 40.000. We do not reset the filter in the case of filter divergence to observe the full behavior of our strategy. Here, we consider only scenario C, since it is the most realistic scenario.

We can see in Fig. 3 that an ambiguity is solved around iteration 600. This can be observed in Fig. 2(c) looking at the two twin galleries connected by a corridor. As the robot leaves the first gallery and enters the corridor, a mode of the distribution hits a wall and vanishes. Afterwards, all particles should converge to the correct solution. However, this does not always happen with large kernels, as can be seen in Figs. 3(a) and 3(b). The problem is that the density estimate method does not allow reachable unvisited cells inside the kernel profile (centered in the robot). That said, the high number of gaps and doors in this map results in a challenge to large kernels, since the observation model erases information of distant cells in order to preserve local consistency of the online constructed map. The best results were presented by the multi-kernel approaches as we can see in Fig. 3(d).

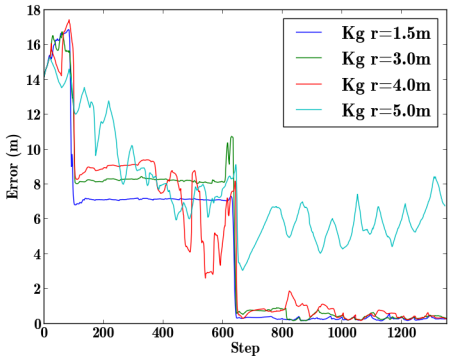
Regarding the computation time, each filter iteration of the approaches using single kernel profiles (Figs. 3(a), 3(b) and 3(c)) was performed in around $30ms$. As expected, the mean computation time for approaches combining three kernels (Fig. 3(d)) was three times larger, around $90ms$. It is important to note that those measurements consider 40.000 particles.

B. Comparing the observation model with other techniques

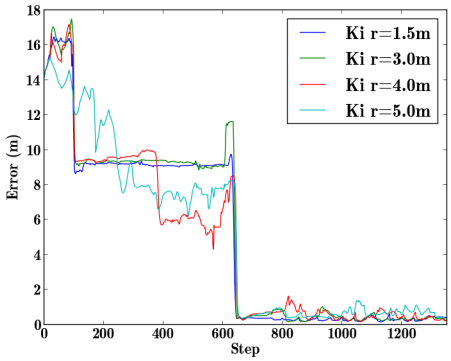
In this section, we compare the proposed observation model with two strategies. The first is an adaptation of the basic beam observation model (**bb**, for short) described by Fox et al. [9], which samples beams spaced by $10°$. We do not pre-compute selected orientations, since this may lead to incorrect observations depending on the angle discretization steps. Moreover, this method can lead to an unreasonable amount of memory. For instance, consider a map having 2.000×2.000 cells, the size of our grid, 8 from 181 sensor beams and five degrees of granularity for the rotation dimension. In this case, such approach would lead to more



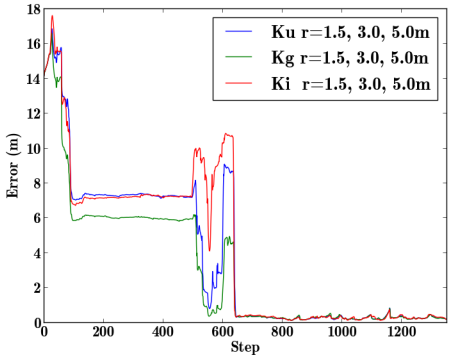
(a) $\mathbf{K_u}$



(b) $\mathbf{K_g}$



(c) $\mathbf{K_i}$



(d) Multi-radius

Fig. 3. Localization in Scenario C using different kernels profiles and radii. Note that in (d) we use homogeneous combinations of kernel profiles having different radius – $1.5m$, $3.0m$ and $5.0m$.

than 8GB of memory. In our case, pre-caching gradients and density estimates for each cell leads to approximately 91MB.

The second strategy is the cosine similarity measure between two sets of readings. We consider the set of beams as an N-dimensional vector, where each dimension corresponds to an angle and has the measured distance as value. Thus, for each particle, we compute the cosine distance between the normalized vector $\mathbf{v}_r$ associated to the robot and the normalized vector $\mathbf{v}_p$ associated to the particle. The idea is that the smallest the angle between two vectors more similar they are. This measure is efficiently computed through the dot product of the vectors.

$$\tau(\mathbf{v}_r, \mathbf{v}_p) = \mathbf{v}_r.\mathbf{v}_p \tag{11}$$

In practice, we modify Eq. 11 to obtain a faster decay of the function associated to the particles weights.

$$\tau_s(\mathbf{v}_r, \mathbf{v}_p) = (\tau(\mathbf{v}_r, \mathbf{v}_p))^4 \tag{12}$$

In order to compare all techniques, experiments were performed in Scenarios A, B and C, varying the number of particles. We performed 15 runs in each scenario for each technique and computed the mean error *per* step as well as the mean time *per* iteration step (see Fig. 4). In all scenarios, $\mathbf{bb}$ and τ were compared with three homogeneous combinations of kernel profiles $\mathbf{K_u}$, $\mathbf{K_g}$, and $\mathbf{K_i}$, all of them with kernels of radius $\mathbf{r} = \{1.5m, 3.0m, 5.0m\}$. Both $\mathbf{bb}$ and τ_s were tested with 500 particles in Scenario A, 1000 particles in Scenario B and 2000 particles in Scenario C, while our proposed techniques were tested with 20000 particles in Scenario A, 30000 particles in Scenario B and 40000 particles in Scenario C.

As we can see, despite the large differences in the number of particles, it is clear from Figs. 4(b), 4(d), 4(f) that the proposed approach is considerably faster than the competing methods. Also, as expected, the increase in the number of particles from 500 to 1000 and 2000, implies linear increases in the mean computation time of the competing methods. Thus, substantially increasing the number of particles can be problematic for both competing methods. On the other hand, the proposed observation model does not suffer with such problem. In fact, the change in the number of particles from 20000 to 30000 and 40000, resulted in modest changes in the mean computation time. Note that, even though the observation model can use kernels of considerably large radii and a large amount of particles, the overall computation time is considerably small. The KDE surrounding the robot is computed only once, what makes the kernel size computationally inexpensive when compared with the cost of weighting and resampling all particles according to their pre-computed density values.

Figs. 4(a), 4(c), and 4(e) present the mean error per step considering 15 runs of each observation model. In scenario A, all techniques converge to the correct robot position at most after step $\approx$ 1.100. We can see that the τ_s method (cyan) converges faster than $\mathbf{bb}$ (purple). However, the 40 times larger set of particles of the proposed method result in much faster convergence (around step 100).

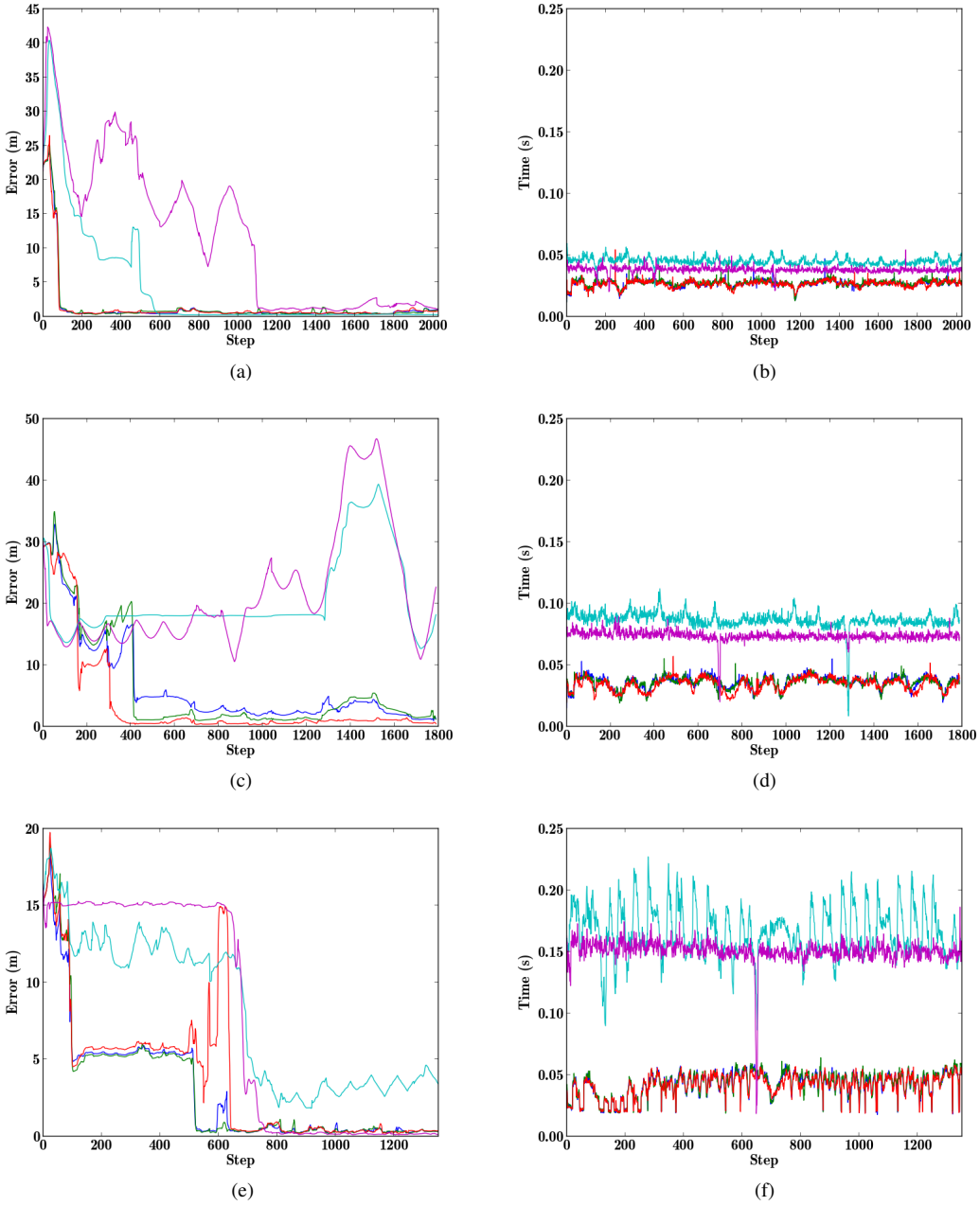


Fig. 4. Mean errors (a, c and e) and mean times per iteration (b, d and f) for scenarios A, B and C, respectively, displaying: $\mathbf{bb}$ (purple), τ_s (cyan), and the three homogeneous combinations of kernel profiles $\mathbf{K_u}$ (dark blue), $\mathbf{K_g}$ (green), and $\mathbf{K_i}$ (red) presented in Section IV-A using a set of radii $\mathbf{r} = \{1.5m, 3.0m, 5.0m\}$. Observe that our technique presents considerably faster times

Scenario B, the largest one, renders weak results for the competing methods, meaning that 1.000 particles is probably not enough to resolve the global localization problem in such map. The degree of ambiguity in such scenario easily makes particles diverge. For instance, we can see that this time, τ_s method diverges and keeps a wrong solution almost until the end. Interestingly, the correct solution and the particle mode maintain the same (incorrect) distance between each other for quite some time. The kernel-based techniques were the only ones to properly converge to the correct solution, maintaining a computation time about 50% smaller than the competing methods. An explanation is that Scenario B is not entirely symmetric, in fact, the width of the corridors varies at most $5cm$. Such small variations do not strongly impact the results of beam-based models, but impact the density estimates. Given that the density computation is related to the number of free-space cells in a grid, small variations in the size of the free-space may imply addition/exclusion of full rows of cells. From the three profiles of kernels tested, the Inverted kernel $\mathbf{K_i}$ (shown in red) presents the best results, and this is expected because such kernel gives more importance to the boundary cells of the kernel, i.e. the

ones more affected by addition/exclusion of rows of cells.

The situation is similar for scenario C, where we see bb converging to the wrong particle and keeping them until ambiguity is eliminated by the motion model near step 600. The τ_s method cannot properly converge even at the end of the process. Once again, the results of the proposed method are the ones with the smaller error.

V. Conclusion

In this paper we have proposed a strategy for Monte Carlo Localization based on kernel density estimates of the free space surrounding the robot. As the experiments have shown, the computation time of our technique is remarkably low in comparison with other techniques, but obtaining similar and better quality of results.

The main advantages of our observation model are:

(i) orientation independence – each position in space can be associated to a single value, disregarding the orientation of the robot;

(ii) a strategy to cut out particles in incorrect directions using the gradient of the scalar field of the density estimates map;

(iii) low time cost - the computation time associated to the particles weighting corresponds to a table look-up;

(iv) low memory consumption – differently from approaches such as Fox et al. [9], in which pre-caching adds a considerable memory burden to the process, the pre-caching of a kernel density information implies the addition of only one value per cell of the grid.

Note that the proposed method performs a dimensionality reduction of all sensors readings. While this allows all the benefits mentioned above, we highlight that it decreases its effectiveness. This is compensated by a massive increase in the number of particles, with minimum impact on processing time.

Our method also depends on visible cells surrounding the robot. For instance, when the robot is approaching corners and bifurcations, the model is unable to stablish a valid observation until the current local map is complete – at least in the surroundings of the robot. If we always use the momentaneous local map, correspondence problems may arise. For example, looking back at Fig. 1 (t=1), and supposing that the robot is pointing upwards (and not downwards as in the figure). Then, from the position of the robot, part of the green area, which contributes to the computation of the kernel density estimate, is unobservable until the robot actually enters the horizontal corridor. This may cause large disparities between what the robot currently observes and what the stored map reports. To circumvent this problem, our method avoids penalizing particles in such situation, thus, increasing the uncertainty of the filter.

The aforementioned characteristics make our method ideal for global localization. In particular, it is suited to reach the tracking stage in large maps, where a large number of hypothesis must be tested at once. After that, another strategy can be employed with a reduced number of particles to improve localization precision.

In the future, we plan to try different combinations of kernels and also different kernel profiles to improve localizability results. Also, it is possible to combine the proposed observation model with others to achieve better localization results once the rough position of the robot is found – e.g. changing to a more accurate and computationally expensive observation model with less particles.

Acknowledgment

Special thanks to Jessica Daltrozo and Mariane Giambastiani for the support. And also to the reviewers, who provided outstanding contributions to improving the quality of this manuscript.

References

[1] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte, "An experiment in integrated exploration," in *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1. Piscataway, NJ, USA: IEEE Press, 2002, pp. 534–539.

[2] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An atlas framework for scalable mapping," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2. Piscataway, NJ, USA: IEEE Press, sept. 2003, pp. 1899–1906.

[3] R. Maffei, V. Jorge, M. Kolberg, and E. Prestes, "Segmented dp-slam," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 31–36.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*, ser. Intelligent robotics and autonomous agents. Cambridge, MA, USA: MIT Press, aug 2005. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262201623

[5] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 376–382, Jun 1991.

[6] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1998.

[7] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA)*. Piscataway, NJ, USA: IEEE Press, May 1999.

[8] I. J. Cox and J. J. Leonard, "Modeling a dynamic environment using a bayesian multiple hypothesis approach," *Artificial Intelligence*, vol. 66, no. 2, pp. 311 – 344, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370294900299

[9] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[10] S. Thrun, "A probabilistic on-line mapping algorithm for teams of mobile robots," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.

[11] T. Duckett and U. Nehmzow, "Mobile robot self-localisation using occupancy histograms and a mixture of gaussian location hypotheses," *Robotics and Autonomous Systems*, vol. 34, no. 23, pp. 117 – 129, 2001, european Workshop on Advanced Mobile Robots.

[12] L. Zhang, R. Zapata, and P. Lpinay, "Self-adaptive monte carlo localization for mobile robots using range finders," *Robotica*, vol. 30, pp. 229–244, 3 2012.

[13] R. R. Murphy, *An Introduction to AI Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, nov 2000. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0262133830

[14] L. Liu, D. Zhang, and J. You, "Detecting wide lines using isotropic nonlinear filtering," *Image Processing, IEEE Transactions on*, vol. 16, no. 6, pp. 1584–1595, 2007.