Map As the Hidden Sensor: Fast Odometry-Based Global Localization

Cheng Peng[1,2] and David Weikersdorfer[1]

Abstract— **Accurate and robust global localization is essential to robotics applications. We propose a novel global localization method that employs the map traversability as a hidden observation. The resulting map-corrected odometry localization is able to provide an accurate belief tensor of the robot state. Our method can be used for blind robots in dark or highly reflective areas. In contrast to odometry drift in the long-term, our method using only odometry and the map converges in long-term. Our method can also be integrated with other sensors to boost the localization performance. The algorithm does not have any initial state assumption and tracks all possible robot states at all times. Therefore, our method is global and is robust in the event of ambiguous observations. We parallel each step of our algorithm such that it can be performed in real-time (up to ~ 300 Hz) using GPU. We validate our algorithm in different publicly available floor-plans and show that it is able to converge to the ground truth fast while being robust to ambiguities.**

I. Introduction

Accurate and robust self-localization is one of the fundamental tasks for indoor robots. If a robot is provided with a false location, it can be catastrophic to other core functionalities such as planning and navigation. Most localization methods integrate odometry information generated by differential wheels or inertial measurement unit (IMU). The advantage of using odometry is the local consistency. The trajectory is guaranteed to be smooth, since it is built by accumulating odometry data over time. However, the trajectory will drift due to error accumulation. To improve the localization accuracy, prior works generalize and fuse other sensors which have descriptive observations. The observations from different sensors such as visual features [2]–[5], 2D/3D point clouds [6][7] from cameras, and LIDAR sensors are combined with odometry information to correct the drift and provide efficient localization and tracking [8]–[14]. However, such vision-based auxiliary features are not always available. For example, visual features can fail in dark environments and LIDAR features can fail in heavily reflective environments.

In this paper, we revisit the fundamental question: if only a sequence of odometry data with an environment map (floor plan) are provided, can the robot still be able to localize itself? In another word, we study whether a blind robot can efficiently localize itself. It is observed that the trajectory integrated from a sequence of odometry data may penetrate through walls and obstacles (shown in Fig 4) due to noise

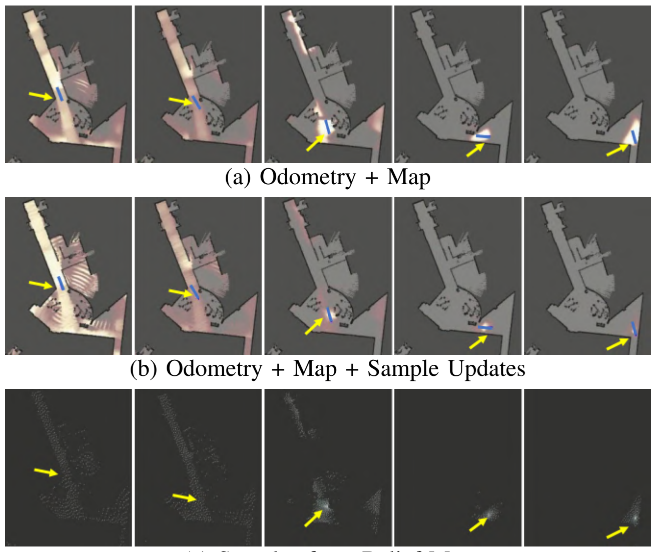

(a) Odometry + Map
(b) Odometry + Map + Sample Updates
(c) Samples from Belief Map

Fig. 1: The changes of the belief map through time (from left to right). Yellow arrows point to the ground truth in blue. (a) The belief map using only odometry with map-correction. (b) The belief map with LIDAR sample updates which converges faster than the previous version. (c) The robot state samples using the Floyd Steinberg method [1].

and drift, which is physically impossible. Our key conjecture is that the map can be exploited as an additional sensor, and the odometry propagation should be corrected by the traversability of the map. As a proof of concept, we focus on indoor robot localization with only odometry and a map as inputs. We further illustrate that our method can be fused with other sensors if available to boost the performance.

We propose a method to incrementally merge odometry data with the guidance from the map. Our method generates a global state belief *tensor* in real-time that not only incorporates multiple location and orientation hypotheses but also tolerates the motion uncertainty from the odometry data. The belief tensor is a dense representation such that all discrete robot states can be estimated in real-time, which has never been fully explored before. Comparing to the sampling methods [14]–[16] that are semi-global and cannot prevent localization failure (due to particle depletion, symmetry, and lack of features [2], [3], [6], [7]), our dense belief tensor guarantees that the correct robot state is always updated and is within high probability region. While other auxiliary sensors such as camera or LIDAR could be misleading due to similar appearance, our belief map is able to maintain multiple hypotheses of the possible robot state to prevent a catastrophic localization error. Using such robust and

*This work was supported by NVIDIA

[1]Cheng Peng and [1]David Weikersdorfer are with ISAAC SDK team, NVIDIA Corporation, Santa Clara, CA 95051

[2]Cheng Peng is with the Department of Computer Science, University of Minnesota, Twin Cities, Minneapolis, MN, 55455 peng0175@umn.edu

global location prior, we can also efficiently recover from localization failure.

Our core contributions and novelties are the followings. Firstly, we propose a complete formulation of the map-corrected odometry update using recursive Bayesian method [17] that provides robust guarantees. Secondly, we parallel each step of our algorithm such that it can be performed in real-time (up to $\sim$ 300Hz). Lastly, we construct our algorithm such that the belief tensor can be fused with other sensor updates (with constraint in Eq 11) to improve the localization accuracy. We test our algorithm extensively in several indoor environments. We also show that even in the existence of topological ambiguities, our algorithm is still able to converge to the exact location by paying a short traveling cost.

II. Related Work

A. Bayesian filtering approach

One of the fundamental methods to localize robot positions is the Bayesian filter method [8], [9], [14], [18]. One of the approaches called Extended Kalman Filter (EKF) [19], [20] assumes the observation and motion models of the robots are Gaussian distributions. They have proven to be very efficient to track the locations and merge the interactions with landmarks. However, it is unable to track multiple hypotheses for posterior distributions that are non-Gaussian, which could arise when the robot is lost or ambiguous observations are made.

Another approach [14] attempts to solve the arbitrary state representation using Monte Carlo sampling. The posterior state distribution is represented as a sum of discrete samples. It is also known as the *particle filter* [21]. Comparing to the EKF method, each particle is a robot state with weight associated with the corresponding probability density. Instead of the analytical results obtained from EKF, particle filer focuses on numerical integration. It allows for simple implementation without any linearity or Gaussian models. Despite the advantages, efficient and representative sampling of the posterior distribution is not guaranteed [22]. Sampling bias will be introduced when the target and the proposal distributions are largely different, which is known as the particle depletion problem [15], [23]. It is due to the underlining Markov model properties [18] where the wrong location can be identified as a sink and thus the system is unable to recover from such localization failure. There are some works [17], [18], [24] that propose to resample the entire state space when localization failure occurs so that the particles can re-center themselves at the correct location. However, the resampling process does not necessarily guarantee a state close to the ground truth and have to rely on localization failure detection methods [25], [26], which is not trivial itself. On the contrary, our method tracks all discrete states at all time and guarantees that the correct trajectory always survives even in the event of localization failure.

Another disadvantage is that the particles are distinct and they can be re-sampled extensively with similar score, which wastes computational power. It is because their states

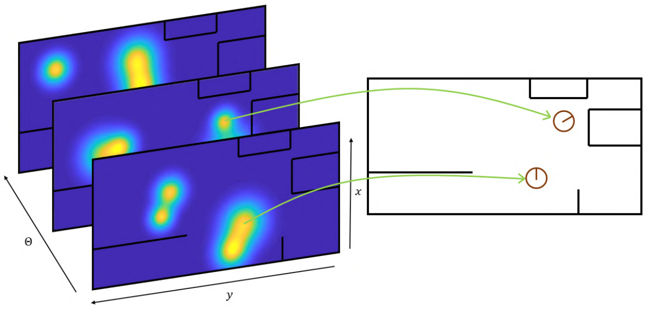


Fig. 2: Illustration of the belief tensor where each channel represents different robot orientation.

cannot be merged and have to be represented discretely. In comparison, our idea is similar to an old work from Burgard et. al [27] and [17] called histogram filter, which has the advantages that all states can be discretely represented as grids in the map. The state probabilities can flow through the grids, which eliminates the necessity to sample and resample.

B. Map integration

Other works [16], [28]–[30] explore the idea of using map as additional constraints to robot motion on top of particle filters. A particle is considered valid only if the corresponding trajectory does not intersect the obstacle map. But their methods does not provide a complete formulation to integrate with map nor any guarantees to the localization capabilities.

Using map information for localization has another set of approaches call *map matching* [31]–[34]. Romero et. al [31] proposed a novel method that integrates the map information. The observation is that odometry is relatively accurate in the short terms and the corresponding trajectory segment is unique in certain region of the map such as turning points. By identifying those unqiue regions through short trajectory segments, localization accuracy can be improved. In contrast, our proposed algorithm can utilize the occupancy map to correct the odometry drift at any location, which provides a much stronger prior to localization. Since we only constrain the state to be within a traversable region, the information from unique turning regions can be integrated as well as other regions such as a straight hallway.

Instead of representing the world as occupancy grids, another method [35] uses topological road maps to provide guidance to odometry for localization on the highway. Using the angular deviation between the road and the vehicle, it is able to successfully locate itself after a few unique turns. This work targets outdoor localization. However, highway road structure is very unique since the vehicle must follow the road and cannot diverge to any open space. Comparing to indoor localization, if only odometry is available, large open space will be a tremendous challenge.

III. Method overview

In our method, the state uncertainty distribution is updated with an occupancy map $\mathcal{M}$, where a pixel region m is characterized as free $P(m_{free} \in \mathcal{M}) = 1$ or occupied space $P(m_{occ} \in \mathcal{M}) = 0$. The state of the robot at time t is $x_t \in SE(2)$ that contains both position and orientation. Each pixel will represent a state of the robot and we assume a uniform distribution for each state $\hat{x}_t$ within a predefined pixel region q.

$$P(x_{q,t}) = |q|^{-1} \int_q P(\hat{x}_t) d\hat{x}_t \qquad (1)$$

where $|q|$ is the area of the region q. The odometry motion update at time t is defined as u_t such that the transitional probability from state x_t to x_{t+1} is given as $P(x_{t+1}|u_t, x_t)$. A observation from an auxiliary sensor is defined as z_t and the probability of the observation is $P(z_t|x_{q,t})$.

A. *Belief Tensor*

Since the state of the robot is initially unknown, we assume all free space in the map are possible robot locations. We construct a dense belief tensor $\mathcal{B}_t : W \times H \times \Theta$ at time t where $W \times H$ is the width and height of the occupancy map and Θ is the orientation as shown in Fig 2. A location $[i, j, k]$ in the belief tensor represents a robot state $x_t = [i\delta_W, j\delta_H, k\delta_\Theta + \theta_t]^T$ where δ_W, δ_H, and δ_Θ is the resolution of $\mathcal{B}$ and k starts from zero. θ_t is the current rotational angle with respect to the initial state.

1) Motion update: We update each channel of the tensor independently with the movement $u_t = [u, v, w]^t$. For a channel k, the corresponding motion is defined as the following.

$$u_{t,k} = R_{x_t} u_t(1:2) \qquad (2)$$

where R_{x_t} is the rotation matrix as

$$R_{x_t} = \begin{bmatrix} \cos(k\delta_\Theta + \theta_t) & -\sin(k\delta_\Theta + \theta_t) \\ \sin(k\delta_\Theta + \theta_t) & \cos(k\delta_\Theta + \theta_t) \end{bmatrix}$$

and $u_t(1:2) = [u, v]^T$. The specific motion update for each channel can be done using an affine transformation with matrix $[I_{2\times2}, u_{t,k}]$, where $I_{2\times2}$ is a 2×2 identity matrix. The resulting belief tensor is

$$\mathcal{B}'_t = Affine(\mathcal{B}_t, u_t) \qquad (3)$$

where $Affine(\cdot)$ applies affine transformation of $[I_{2\times2}, u_{t,k}]$ to each channel k in $\mathcal{B}_t$.

The internal orientation is then updated as

$$\theta_{t+1} = \theta_t + u_t(3) \qquad (4)$$

where $u_t(1) = w$.

2) Motion uncertainty: For a discrete region q, the probability that the robot state is at location q is given as

$$P(x_{q,t+1}|u_{1:t}) = \sum_{i\in\mathcal{M}} P(x_q|u_t, x_{i,t}) P(x_{i,t}) \qquad (5)$$

where $u_{1:t}$ is all the observations from time 1 to t.

Using Equation 5, we can update the probability of each state in the tensor by aggregating the surrounding values. Assuming our motion uncertainty is a Gaussian distribution with covariance matrix defined as $\Sigma_t \in R^{3\times3}$, the Gaussian kernel is the expressed as

$$G(x_t, \Sigma_t) = \exp(-\frac{1}{2}(\overline{x_t} - x_t)^T \Sigma_t^{-1} (\overline{x_t} - x_t)) \qquad (6)$$

for the current state x_t and its surrounding location $\overline{x_t}$. The covariance matrix Σ_t is defined as

$$\Sigma_t = R_z(x_t)\ diag(\sigma_x^2, \sigma_y^2, \sigma_\theta^2) R_z(x_t)^T \qquad (7)$$

where $R_z(x_t) \in R^{3\times3}$ is the rotation along the z axis with the angle $k\delta_\theta + \Theta_t$ and $diag(\cdot)$ is the diagonal matrix of a vector.

We can generalize the operation as

$$\mathcal{B}''_{t,k} = Conv(\mathcal{B}'_t, G(x_t, \Sigma_t)) \qquad (8)$$

where the function $Conv(\cdot)$ is a 3D convolution function using kernel $G(x_t, \Sigma_t)$.

As shown in Fig 2, each channel is a belief map representing the state probability with different orientation.

3) Map sensor update: Each time after applying the affine transformation and motion blur (Eq. 3 and Eq 8), the final "observation" update of odometry-only belief tensor $\mathcal{B}''_{t,k}$ is multiplied with the map probability $\mathcal{M}$. Since $P(m_{occ} \in \mathcal{M}) = 0$ for obstacles and $P(m_{free} \in \mathcal{M}) = 1$ for free space, the probability in obstacle locations will be reset to zeros. The initial state assumption and the trajectory that leads to the obstacles is then an invalid assumption.

We show the detailed algorithm in the pseudo-code below, where $\odot$ is pixel-wise multiplication and N^{-1} is the pixel-wise inverse of matrix N.

Algorithm 1 Belief tensor update

```
Require: $\mathcal{B}_0$
1: if $|u_{t,k}| >= 1$ then
2:     for $k = 0$ to $|\Theta|$ do
3:         $\mathcal{B}'_{t,k} = Affine(\mathcal{B}_{t,k}, u_{t,k})$
4:         $\mathcal{B}'_{t,k} = \mathcal{M} \odot \mathcal{B}'_{t,k}$
5:         $\mathcal{B}''_{t,k} = Conv(\mathcal{B}'_t, G(x_t, \Sigma_t))$
6:         $\mathcal{B}''_{t,k} = \mathcal{M} \odot \mathcal{B}''_{t,k}$
7:         $N = Conv(\mathcal{M}, G(x_t, \Sigma_t))$
8:         $\mathcal{B}_{t+1,k} = \mathcal{B}''_{t,k} \odot N^{-1}$
9:     end for
10: end if
```

The sensor update needs to be done not only after the entire process at line 6 but also in between motion update and uncertainty update at line 4. It is to avoid the convolution process to extract nearby obstacle state that are non-zero due to previous motion updates.

When the robot travels near a wall, the boundary state probability will decrease much faster comparing to traveling in open space. It is because the state probability are simply removed and a Gaussian kernel (Eq 6) may extract zero values from the obstacle regions. It is equivalent to the robot moving along side a cliff instead of a wall, which it may fall into. Therefore, we add line 7 and line 8 to normalize

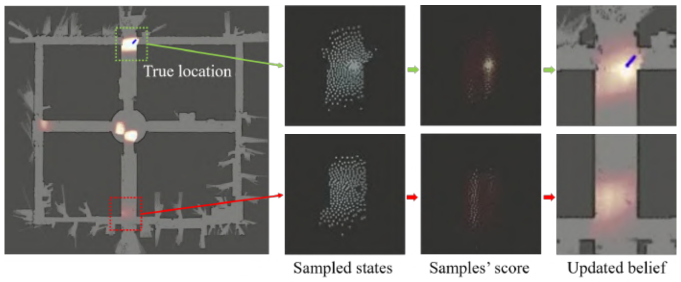


Fig. 3: The ground truth location is supported (brighter) by sample updates while the false location diminish (dimmer) with updates.

the convoluted belief map by the sum of kernel activation as follows.

$$\mathcal{B}_{t+1,k} = \mathcal{B}''_{t,k} \odot Conv(\mathcal{M}, G(x_t, \Sigma_t))^{-1} \tag{9}$$

Since $\mathcal{M}$ is a binary tensor with ones in free space, the values from $Conv(\mathcal{M}, G(x_t, \Sigma_t))$ is just the kernel activation sum for each pixel.

The belief tensor can be converted into a belief map of the environment by taking the maximum value across the channels for each state. The estimates of the robot state is then the maximum probability location in the belief map.

B. Observation update

With the belief tensor of the robot state, it is also possible to update the tensor with observations from other sensors such as LIDAR or camera. However, it is expensive to compare measurements at all locations to update the tensor values. Therefore, we sample a subset of possible states and update the corresponding belief value as follows.

$$P(x_{q,t}|u_{1:t}, z_t) = \eta P(z_t|x_{q,t})P(x_{q,t}|u_{1:t}) \tag{10}$$

where η is the normalization factor such that $\sum_q P(x_{q,t}|u_{1:t}, z_t) = 1$. Using our previously estimated belief map, we can efficiently select a subset of samples that is representative of the belief map distribution.

1) Floyd Steinberg Dithering: To find a subset of samples that follows the probabilistic distribution of the belief map, we borrow the idea from an image dithering technique call Floyd Steinberg Dithering [1]. It is initially introduced as a dithering method to sub-sample the image, which is efficient and simple to implement. Instead of updating all possible locations in the belief tensor, a subset of points are selected to be updated. As shown in Fig 1(c) and Fig 3, we extract samples from the belief map and update them using Eq 10.

2) Integration with other sensors: Since the belief tensor is created with only odometry and an occupancy map, with the additional sampling methods, we can integrate observation update with arbitrary sensors such as LIDAR or camera. To ensure the localization robustness, the sensor observation model has to satisfy the following equation.

$$P(z_t|x_t^*) \geq P(z_t|x_t) \tag{11}$$

where x_t^* is the ground truth location and $x_t \neq x_t^*$ is any other location at time t. It is important to note that symmetry and lack of features have been significant challenges to location detection [2], [3], [6], [7]. However, our method relaxes the constraint such that the posterior of the ground truth location $P(x_t^*|u_{1:t}, z_t)$ does not disappear in the event of confused observations. It means that as long as the observations from other similar looking locations are not better than that of the correct location, the accuracy is guaranteed.

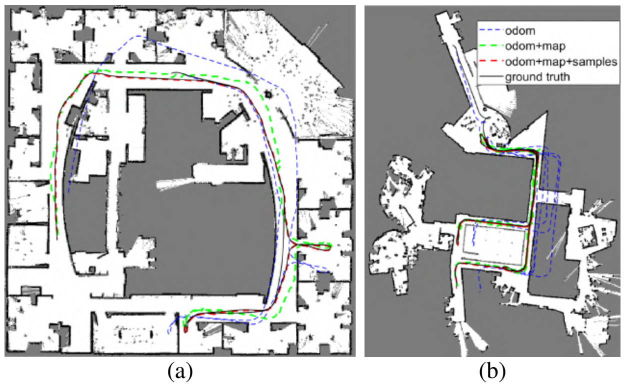


Fig. 4: Comparisons among ground truth trajectory to our method with and without sample updates. The odometry will drift over time but odometry + map will stay within the vicinity of the ground truth location. With LIDAR sample updates, the final trajectory overlaps with the ground truth.

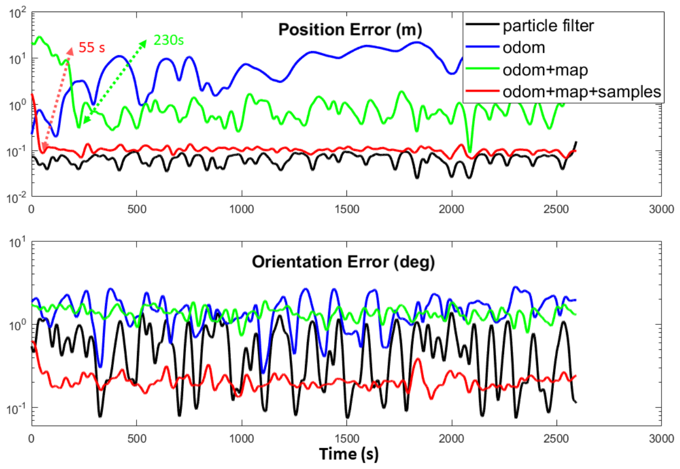


Fig. 5: The long-term localization error comparison. The position error converges after 230 seconds for our method w/o sample updates and 55 seconds for our method w/ sample updates.

IV. Experiments

To evaluate our localization method, we conduct experiments in different environments from a public data set [36]. The algorithm is written in c++ and runs on an Ubuntu system with an i7 Intel core and a Titan V graphic card. The simulation environment is from ISAAC SDK [37] which provides realistic LIDAR sensor measurements for any given floor plans. We compare our algorithms with a particle filter method [15] that starts at the highest $P(z_0|x_0)$ for all $x_o \in \mathcal{M}$. A total of 75 particles are used to track the local state from the estimated initial location. In this section, we show the robustness of our algorithm, by real experiments, in terms of (1) fast and guaranteed convergence at the location where prior methods have ambiguities, (2) accuracy and stability in long term localization task, and (3) the capability to be fused with other sensors for performance improvement.

The particle filter runs at ~ 20 ms and our method runs at ~ 3.2 ms on average without sample updates. With sample updates from LIDAR sensor, the average run-time is ~ 31.2 ms. Our method achieves real-time performance using a belief tensor of size $500 \times 500 \times 128$ (depends on the actual map size), with pixel resolution of 0.1 meters and angular resolution of $2.81°$ respectively.

A. *Information from the map sensor*

To demonstrate the information collected from the "map sensor", we show the qualitative results in Fig 4. The initial estimates are discarded, since the robot needs to move around (100 meters) to localize itself. As is shown in Fig 4, The odometry trajectory drifts from the ground truth trajectory. The trajectory from the map-corrected odometry shows no large divergences and stays in the vicinity of the ground truth trajectory, though not quite accurate. It is expected since any precise location information is not given. After fusing with the LIDAR observation, the final trajectory overlays with the ground truth trajectory. Fig 5 shows the quantitative results where the odometry + map trajectory stays within around 1 meter accuracy to the ground truth after converging.

B. *Localization robustness*

Previous methods do not track dense robot states at all times due to computational limitations. They assume that the initial location is known or is at the region with the highest sensor response, and start tracking following only local information. Since there is only one ground truth, a tracking method that assumes locally distributed noise is used. However, such mechanism may fail in the existence of observation ambiguity due to topology symmetry. Fig 3 shows an example of narrow corridors where localization methods based on LIDAR observation encounter confusion.

To characterize the localization difficulty in different maps, we evaluate the localization error of LIDAR observations at each possible robot location. More specifically, for each location $p = [x, y]$ with observation z, we estimate the location $\hat{p} = [\hat{x}, \hat{y}]$ that best matches z. In perfect setting, $p = \hat{p}$, but it is not always true in the existence of noise. The localization error is then defined as $e_p = ||p - \hat{p}||_2$ for each location p. Therefore, the localization difficulty of a map can be estimated as

$$D_{\mathcal{M}} = \frac{\sum_{p \in \mathcal{M}} I(e_p > C)}{\sum_{p \in \mathcal{M}} p} \tag{12}$$

where $I(\cdot)$ is the indicator function that outputs 1 if $e_p > C$, $C = 1$ meter, and $\sum_{p \in \mathcal{M}} p$ is the number of free space in map $\mathcal{M}$.

The wrongly localized column in Table I shows different maps' difficulty levels. Maps with higher difficulty are the

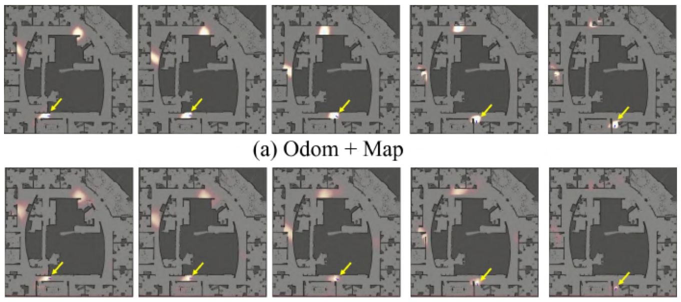


Fig. 6: Both (a) and (b) have more than one high probability regions. As more unique structures are observed in (b), the incorrect locations disappear faster than those in (a).

ones with similar hallways where the LIDAR measurements cannot be distinguished. In the map *ACES3 Austin*, 8.5% wrongly localized regions correspond to more than $28,000$ locations at which the map is ambiguous. Even in the map with the lowest difficulty (*Seattle, UW*), more than 1400 locations can be wrongly localized with at least 1 meter error.

We show that our method is robust to those ambiguous regions and is able to accurately localize the robot by moving less than 50 meters. The robot starts at 500 arbitrary locations including 250 wrongly localized regions for each map. Since the initial location is unknown, the robot need to travel for a certain distance until localized. The robot's travel policy is random walk with no external guidance. Once the localization error is less than 10 cm for more than 1 mins, the robot is considered localized. The distribution of the convergence distance is shown in Table I. There are a few cases where the convergence distances are large (> 200m). They result from the exploration trajectory overlapping itself due to random motion. For example, in the Intel research map shown in Fig 4, the structure tolerates circular robot motion without being able to pin point the exact location.

When ambiguous observations are encountered, our method determine the exact location by moving a short distance that aggregates the unique observations along the trajectory. The belief tensor will always maintain a high probability at the true location as shown in Fig 7. Even though there may exist other incorrect locations, they will not survive for long. Fig 3 and Fig 6 show the cases where there are multiple high probability estimation of the robot location. However, as additional observations and the traversability constraints are imposed, only the true location survives. When auxiliary sensors are used, we can guarantee that the algorithm will not lose track of the ground truth location as long as the constraint from Eq 11 is satisfied.

C. *Long term localization accuracy*

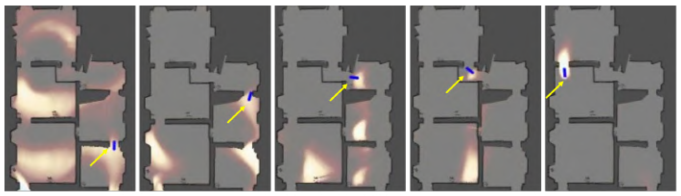
We run our algorithms for 30 mins to test the long term localization error in all maps. Table II shows the accuracy comparison with a traditional particle filter method. It is shown that our method achieves superior performance on orientation accuracy with competitive performance on location accuracy after initial stage of information integration. After 5 mins, our methods without samples can also provide

	min error (m)	median error (m)	max error (m)	wrongly localized ($> 1m$)	Dimension (pixel = 0.1m)
ACES3 Austin	15.3	44.1	132.0	8.5%	586 × 556
Belgioioso Castle	9.9	32.6	105.3	1.2%	993 × 206
MIT CSAIL	9.4	27.9	63.4	1.0%	482 × 668
Intel Research Lab	18.4	93.7	362.5	0.2%	579 × 581
Seattle, UW	4.9	30.8	79.2	0.7%	908 × 229

TABLE I: Localization convergences for different maps.

		0 to 1 mins		1 to 5 mins		5 to 30 mins	
		XY error(m)	θ error(deg)	XY error(m)	θ error(deg)	XY error(m)	θ error(deg)
ACES3 Austin	particle filter	**0.054**	0.757	**0.050**	0.566	**0.062**	0.677
	ours w/o samples	15.339	1.471	0.613	1.407	0.603	1.449
	ours w/ samples	4.913	**0.367**	0.117	**0.204**	0.106	**0.209**
Belgioioso Castle	particle filter	**0.038**	1.102	0.039	0.431	**0.034**	0.358
	ours w/o samples	11.068	1.355	14.713	1.385	0.496	1.422
	ours w/ samples	1.808	**0.648**	**0.101**	**0.239**	0.129	**0.204**
CSAIL MIT	particle filter	**0.038**	**1.102**	**0.039**	0.431	**0.034**	0.358
	ours w/o samples	20.121	1.183	4.997	1.644	0.415	1.466
	ours w/ samples	1.099	1.152	0.200	**0.265**	0.129	**0.252**
Intel Research Lab	particle filter	**0.087**	**0.552**	0.070	0.662	0.072	0.623
	ours w/o samples	10.928	1.462	0.996	1.419	0.965	1.408
	ours w samples	5.325	0.723	**0.066**	**0.243**	**0.065**	**0.208**
Seattle, UW	particle filter	**0.059**	1.033	**0.062**	0.591	**0.065**	0.514
	ours w/o samples	24.356	1.581	4.657	1.454	0.751	1.345
	ours w/ samples	0.548	**0.295**	0.103	**0.205**	0.104	**0.206**

TABLE II: Mean long term localization error of each map within different time intervals.


(a) Odom + Map

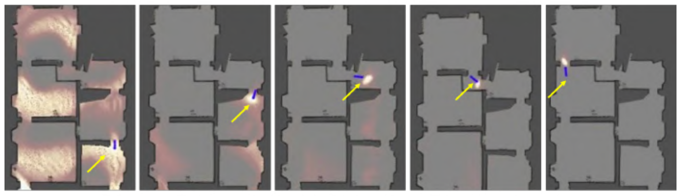
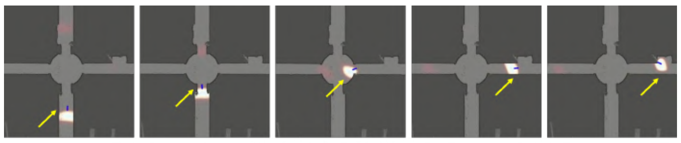

(b) Odom + Map + Samples Update

Fig. 7: While multiple hypotheses survives in (a) initially, the unique motion pin points region that surrounds the ground truth. (b) converges much faster than that in (a) because of the additional observations from LIDAR measurements.

good localization up to 1 meter accuracy. Fig 5 shows one instance for the long term error comparison.

V. Conclusion

In conclusion, we propose a method to use occupancy map as an additional sensor such that a blind robot with only odometry is able to localize itself in real-time. We also adapt our algorithm to other auxiliary sensors (such as LIDAR) efficiently. The resulting algorithm is able to provide a robust belief tensor of the robot state with accurate localization. Since our method constantly tracks dense robot states, it is robust and can recover from localization failure in the event of ambiguous observations. We validate our algorithm in publicly available floor plans which contains difficult scenarios such as symmetric structures. The experiments show that our method can successfully localize the robot despite of the ambiguities. We can also achieve real-time performance (up to $\sim$ 300Hz for odometry + map and $\sim$ 30Hz with LIDAR samples). The convergence distance histogram can be heavily tailed. Therefore, we plan to explore active localization so that the distance traveled until convergence is minimized.


(a) Odom + Map

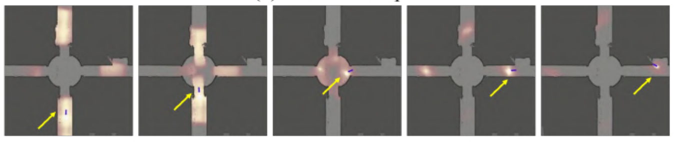

(b) Odom + Map + Samples Update

Fig. 8: While (a) only have one high probability region, (b) has multiple due to similar LIDAR observations in the hallway. Despite the confusion from LIDAR, as the robot travels to a unique location, the region converges again to the ground truth.

Acknowledgement

The authors of this paper would like to thank Dmitry Chichkov, Benjamin Butin, Qian Lin, and Daniel Abretske who provide tremendous help on debugging and accelerating the algorithm. The authors would also like to thank Shan Su for her supports and valuable discussions. Thanks go to Cyrill Stachniss, Dirk Haehnel, and Patrick Beeson for providing the map data [36].

References

[1] Floyd, R.W., Steinberg, L.: An adaptive algorithm for spatial grayscale. In: Proceedings of the Society of Information Display. (1976) 75–77

[2] Gálvez-López, D., Tardós, J.D.: Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics **28**(5) (October 2012) 1188–1197

[3] Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. International journal of computer vision **42**(3) (2001) 145–175

[4] Schönberger, J.L., Pollefeys, M., Geiger, A., Sattler, T.: Semantic visual localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 6896–6906

[5] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2921–2929

[6] Yang, J., Cao, Z., Zhang, Q.: A fast and robust local descriptor for 3d point cloud registration. Information Sciences **346** (2016) 163–179

[7] Dubé, R., Dugas, D., Stumm, E., Nieto, J., Siegwart, R., Cadena, C.: Segmatch: Segment based place recognition in 3d point clouds. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2017) 5266–5272

[8] Hess, W., Kohler, D., Rapp, H., Andor, D.: Real-time loop closure in 2d lidar slam. In: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2016) 1271–1278

[9] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B., et al.: Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: IJCAI. (2003) 1151–1156

[10] Se, S., Lowe, D.G., Little, J.J.: Vision-based global localization and mapping for mobile robots. IEEE Transactions on robotics **21**(3) (2005) 364–375

[11] Grisetti, G., Tipaldi, G.D., Stachniss, C., Burgard, W., Nardi, D.: Fast and accurate slam with rao–blackwellized particle filters. Robotics and Autonomous Systems **55**(1) (2007) 30–38

[12] Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE transactions on robotics **31**(5) (2015) 1147–1163

[13] Forster, C., Pizzoli, M., Scaramuzza, D.: Svo: Fast semi-direct monocular visual odometry. In: 2014 IEEE international conference on robotics and automation (ICRA), IEEE (2014) 15–22

[14] Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: ICRA. Volume 2. (1999) 1322–1328

[15] Gustafsson, F.: Particle filter theory and practice with positioning applications. IEEE Aerospace and Electronic Systems Magazine **25**(7) (2010) 53–82

[16] Davidson, P., Collin, J., Takala, J.: Application of particle filters for indoor positioning using floor plans. In: 2010 Ubiquitous Positioning Indoor Navigation and Location Based Service, IEEE (2010) 1–4

[17] Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics. MIT press (2005)

[18] Grisettiyz, G., Stachniss, C., Burgard, W.: Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In: Proceedings of the 2005 IEEE international conference on robotics and automation, IEEE (2005) 2432–2437

[19] Maybeck, P.S.: Stochastic models, estimation, and control. Volume 3. Academic press (1982)

[20] Dissanayake, M.G., Newman, P., Clark, S., Durrant-Whyte, H.F., Csorba, M.: A solution to the simultaneous localization and map building (slam) problem. IEEE Transactions on robotics and automation **17**(3) (2001) 229–241

[21] Doucet, A., Godsill, S., Andrieu, C.: On sequential monte carlo sampling methods for bayesian filtering. Statistics and computing **10**(3) (2000) 197–208

[22] Aulinas, J., Petillot, Y.R., Salvi, J., Lladó, X.: The slam problem: a survey. CCIA **184**(1) (2008) 363–371

[23] Kwak, N., Kim, I.K., Lee, H.C., Lee, B.H.: Analysis of resampling process for the particle depletion problem in fastslam. In: RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication, IEEE (2007) 200–205

[24] Grisetti, G., Stachniss, C., Burgard, W., et al.: Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE transactions on Robotics **23**(1) (2007) 34

[25] Alsayed, Z., Bresson, G., Verroust-Blondet, A., Nashashibi, F.: Failure detection for laser-based slam in urban and peri-urban environments. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), IEEE (2017) 1–7

[26] Klein, G., Murray, D.: Improving the agility of keyframe-based slam. In: European Conference on Computer Vision, Springer (2008) 802–815

[27] Burgard, W., Fox, D., Hennig, D., Schmidt, T.: Estimating the absolute position of a mobile robot using position probability grids. In: Proc. 13th Nat. Conf. of the American Association for Artificial Intelligence, MIT Press (1996) 896–901

[28] O'Kane, J.M.: Global localization using odometry. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., IEEE (2006) 37–42

[29] Beauregard, S., Klepal, M., et al.: Indoor pdr performance enhancement using minimal map information and particle filters. In: 2008 IEEE/ION Position, Location and Navigation Symposium, IEEE (2008) 141–147

[30] Klepal, M., Pesch, D., et al.: A bayesian approach for rf-based indoor localisation. In: 2007 4th International Symposium on Wireless Communication Systems, IEEE (2007) 133–137

[31] Romero, A.R., Borges, P.V.K., Pfrunder, A., Elfes, A.: Map-aware particle filter for localization. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2018)

[32] Chen, D., Driemel, A., Guibas, L.J., Nguyen, A., Wenk, C.: Approximate map matching with respect to the fréchet distance. In: 2011 Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments (ALENEX), SIAM (2011) 75–83

[33] Luo, A., Chen, S., Xv, B.: Enhanced map-matching algorithm with a hidden markov model for mobile phone positioning. ISPRS International Journal of Geo-Information **6**(11) (2017) 327

[34] OCHIENG, W., QUDDUS, M., NOLAND, R.: Map-matching in complex urban road networks. Brazilian Journal of Cartog-raphy **55**(2) (2003) 1–14

[35] Brubaker, M.A., Geiger, A., Urtasun, R.: Map-based probabilistic visual self-localization. IEEE transactions on pattern analysis and machine intelligence **38**(4) (2015) 652–665

[36] Howard, A., Roy, N.: The robotics data set repository (radish) (2003)

[37] NVIDIA Cooperation: Isaac sdk