Monte Carlo Localization using 3D Texture Maps

Yu Fu, Stephen Tully, George Kantor, and Howie Choset

Abstract— **This paper uses KLD-based (Kullback-Leibler Divergence) Monte Carlo Localization (MCL) to localize a mobile robot in an indoor environment represented by 3D texture maps. A 3D texture map is a simplified model that includes vertical planes with colored texture information associated with each vertical plane. At each time step, a distance measurement and an observed texture from an omnidirectional camera are compared to the expected distance measurement and the expected texture according to each hypothesis of the robot's pose in an MCL framework. Compared to previous implementations of MCL, our proposed approach converges faster than distance-only MCL and localizes the robot more precisely than SIFT-based MCL. We demonstrate this new MCL algorithm for robot localization with experiments in several hallways.**

I. Introduction

Localization is essential for mobile robots navigating in a given environment. With a localization system, a robot is capable of inferring its pose in a map based on observations made with onboard sensing: odometry, proximity sensors, cameras, etc. After determining its pose, a robot can then plan paths to target locations and can avoid known obstacles.

There are two generalized methods for a mobile robot to perform localization in a given map: metric localization [1]–[3] and topological localization [4], [5]. Given a 2D map, the result of metric localization is a position (x, y) and heading θ in the map. For topological localization, on the other hand, the result of localization would specify the place or region where the robot is located. In this paper, we explore a metric localization method that models the environment with 3D texture maps.

Probabilistic methods, such as the extended Kalman filter (EKF) and particle filtering (PF), are important for recursively solving for the most likely estimate during localization [6]. The EKF models the robot's possible pose as a Gaussian distribution while a nonparametric method, such as a particle filter, does not restrict the model to any specific distribution. In addition, a particle filter is well suited for solving the problem of global localization because it is inherently testing multiple hypotheses.

When a robot moves, Monte Carlo Localization (MCL) [7] gradually removes unlikely robot pose hypotheses in a particle filter framework by comparing sensor measurements with expected measurements. According to the given map, proximity sensors [2] or cameras [3], [8] are used to perceive the environment. A texture map is a metric map including environmental appearance. Recent work has demonstrated that a texture map can be generated by either fusing a camera and a 3D laser range finder [9] or using a RGBD sensor, such as Kinect$^{\text{TM}}$ (Microsoft Co., Redmond, WA, USA). However, few methods [10] have discussed the technique of localizing in a texture map.

This paper adopts KLD-based (Kullback-Leibler Divergence) MCL for localizing the robot in a given 3D texture map. The 3D texture map is a simplified model that includes vertical planes and textures associated with each vertical plane. During localization, a measured distance and an observed texture are obtained from an omnidirectional camera and then compared to the expected distance measurement and expected texture for each hypothesis of the robot's pose. The distance is extracted from an image by making certain planar assumptions about the environment. Compared to prior work on MCL, the proposed approach converges faster than a distance-only MCL implementation and localizes the robot more precisely than a SIFT-based MCL approach. In addition, all of the hypotheses are better localized around the true pose using our proposed method than with SIFT-based MCL. This is because the expected observed texture can be generated for each possible pose in the 3D texture map, while expected SIFT features can only be generated from the set of positions that were used to originally map the space.

II. Construction of a 3D Texture Map

A 3D texture map is a 3D geometric model of an environment along with its appearance. In this paper, the indoor environment is modeled as a simplified 3D texture map [11] by using an omnidirectional camera. The simplified 3D texture map consists of vertical planes on flat floors with a pixel-based colored texture associated with each vertical plane. The vertical planes are a basic unit representing walls in structured environments.

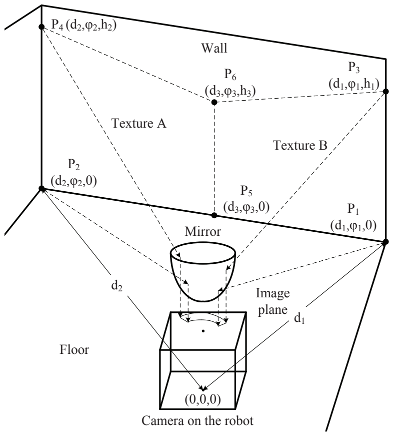
A vertical plane is obtained by detecting the boundary pixels between a wall and the floor in an omnidirectional image and fitting the pixels with a line. By assuming the color change on the boundary is apparent [2], [12], the boundary pixels can be detected by an edge detection algorithm [13]. The relative horizontal distance d between a boundary point and the robot is calculated with a distance sensing function $d = f(\rho)$, where ρ is the image distance between the boundary pixel and the image center. The azimuth angle ϕ of a boundary point can be computed from the coordinate of the

Y. Fu is with the Electrical Engineering Department at National Taiwan University of Science and Technology, Taipei, Taiwan. D9407201@mail.ntust.edu.tw

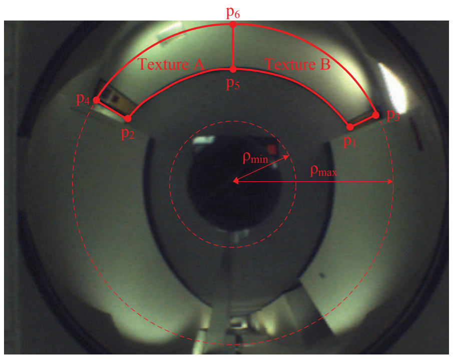
S. Tully is with the Electrical and Computer Engineering Department at Carnegie Mellon University, Pittsburgh, PA 15213, USA. G. Kantor and H. Choset are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213, USA. {stully@ece, kantor@ri, choset@cs}.cmu.edu

corresponding boundary pixel in the image. The positions of all boundary points forms a 2D map. Lines are detected and endpoints are stored as the positions of the vertical planes.

Fig. 1 shows an example for finding the position of a vertical plane. The relative horizontal distance and the azimuth angle of all boundary points on $\overline{P_1P_2}$ in Fig. 1(a) are obtained when all of the boundary pixels on the red arc $\widehat{p_1p_2}$ in Fig. 1(b) are detected.



(a) The projection of texture from a wall onto the image plane.



(b) Pixels projected from texture on an omnidirectional image.

Fig. 1. An overview of our 3D texture map construction.

The observable height of a vertical plane is required in order to correctly map the pixels from an omnidirectional camera to a texture corresponding to the vertical plane because the complete vertical plane may not be observed. Instead of finding the maximal observable height in each azimuth angle, the heights along the same tilt angle $\psi_{\max}$ under different azimuth angles are computed. The tilt angle $\psi_{\max}$ corresponds to a maximal observable view which is represented as an inscribed circle of maximal radius $\rho_{\max}$ centered at the image center, as seen in Fig. 1(b). Since all pixels on the maximal radius $\rho_{\max}$ are projected from points along the same tilt angle $\psi_{\max}$ but in different relative distances, the heights of these points are different. A height sensing function $h = g(d)$ is used to compute the height of a point along the tilt angle $\psi_{\max}$ by giving the relative horizontal distance d. In Fig. 1(a), the maximal observable height of the vertical plane is marked as $\overline{P_3P_6}$ and $\overline{P_4P_6}$.

When the observable heights of a vertical plane and the position of the vertical plane are available, all pixels between the boundary pixels and maximal radius $\rho_{\max}$ are mapped to the vertical plane. Texture above a boundary point is extracted from the omnidirectional image captured on the nearest pose to the boundary point, because a same vertical plane may be projected on many omnidirectional images. The texture mapping from many omnidirectional images to a vertical plane will be detailed in Section II-C.

A. Distance Sensing

The distance sensing function calculates the relative horizontal distance between the robot to a boundary point. In a flat environment, boundary points from different distances are projected on different pixels in the image, as an example shown in Fig. 2.

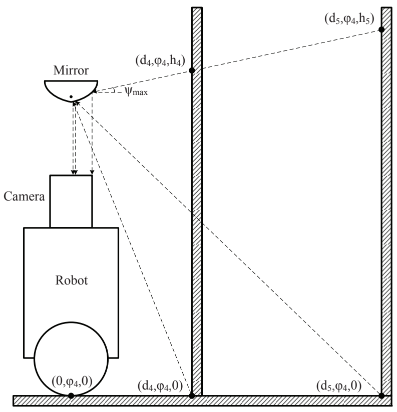


Fig. 2. Distance sensing and height sensing with an omnidirectional camera (side view).

The distance sensing function is obtained by fitting many collected data points which associate true distances with image distances. We chose a 4th order polynomial equation to represent $f(\rho)$ as the selection in [14]. The minimal and maximal measurable distance are 88.9cm and 330.2cm, respectively.

The maximal and minimal measurable distances restrict the location of a boundary pixel within an annulus with inner radius of $\rho_{\min}$ and outer radius of $\rho_{\max}$, as the region between the two dashed circles in Fig. 1-(b).

B. Height Sensing

The height sensing function $h = g(d)$ is used to compute the height of a point along the tilt angle $\psi_{\max}$ by giving the relative horizontal distance d. The height sensing function is a linear function $g(d) = a_0 + a_1 d$, where a_0 is the height of the mirror focus and a_1 is $\tan(\psi_{\max})$ in Fig. 2.

C. Texture Mapping

Instead of blending many textures from images, texture above a boundary point is extracted from the omnidirectional image which is captured at the nearest pose to the boundary point, due to the smallest error between the computed relative horizontal distance and the real distance. Fig. 3 shows an example of texture mapping for a vertical plane from two omnidirectional images. A vertical plane is observed at both pose A and B. Given the boundary points of the vertical plane, as the points along $\overline{P_1^B P_5^B}$, the maximal observable heights of the vertical plane at pose A and pose B are $(\overline{P_{1,A}P_{2,A}}$ and $\overline{P_{2,A}P_{5,A}})$ and $(\overline{P_{1,B}P_{4,B}}$ and $\overline{P_{4,B}P_{5,B}})$, respectively. The textures, as the regions marked with diagonal lines in Fig. 3, above boundary points along $\overline{P_1^B P_3^B}$ and above boundary points along $\overline{P_3^B P_5^B}$ are extracted from the omnidirectional images captured at pose A and on pose B, respectively.

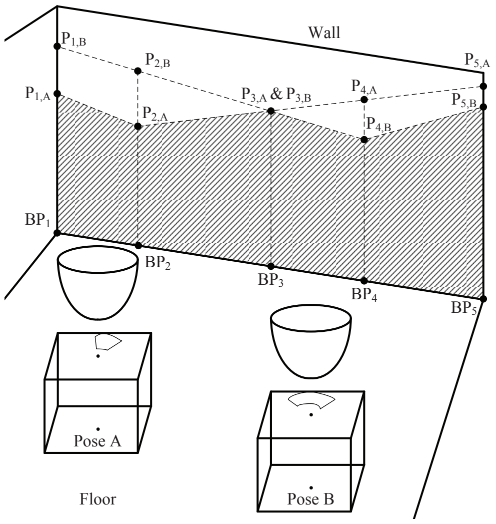


Fig. 3. Texture mapping of a vertical plane from two omnidirectional images.

Fig. 4 shows the constructed 3D texture map on Floor A of Newell-Simon Hall (NSH) at Carnegie Mellon University. White walls and brown doors are two components in this environment. The black part on the top includes the unmapped region of the textures for the vertical planes and the ceiling. The gray part on the bottom is the carpet.

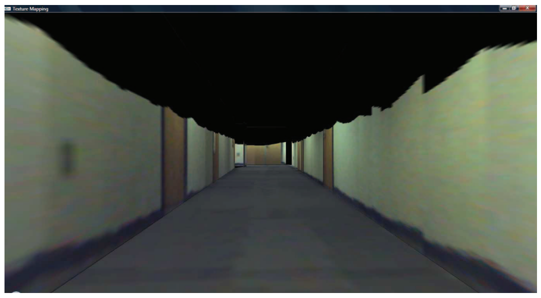


Fig. 4. This is an example of a textured 3D map constructed by a mobile robot with an omnidirectional camera.

III. KLD-Based Monte Carlo Localization in a 3D Texture Map

MCL [7] is a typical approach for mobile robot global localization based on a particle filter. Each particle in MCL represents a hypothesis for the robot's pose: (x, y, θ). After the robot moves, each particle is predicted based on a motion model with the control input. The expected measurement from the predicted pose of each particle is then compared to an actual measurement to decide the weight based on a measurement model. Particles are resampled according to the weights and the above procedure is repeated. Eventually, the particles converge to a region around the true robot's pose if the motion and measurement models are designed properly.

A large amount of particles are typically used when the environment is large in order to have at least one particle initialized near the robot's true pose. Since large numbers of particles become computationally intractable, KLD [6] is used to adaptively reduce the number of particles in order to speed up MCL.

A. Motion Model

The motion model describes the relationship between odometry and the robot's true motion. A 10% error in odometry is assumed for the noise in the motion model, as follows,

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \mathcal{N}(\Delta x, 0.01\Delta x^2) \\ y(k) + \mathcal{N}(\Delta y, 0.01\Delta y^2) \\ \theta(k) + \mathcal{N}(\Delta\theta, 0.01\Delta\theta^2) \end{bmatrix}, \quad (1)$$

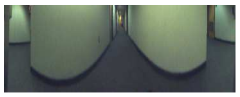
where $\mathcal{N}(\mu, \sigma^2)$ stands for the normal distribution with mean μ and standard deviation σ.

B. Measurement Model

The similarity between two measurements is computed in the measurement model. The weight of a particle is decided by the similarity between the expected measurement of a particle and the true measurement.

Distance measurements and the observed texture are used to localize the robot in the 3D texture map. The weight of each particle can be computed by $weight = \xi_d \xi_t^{K_{dt}}$, where ξ_d is the similarity of the distance measurements, ξ_t is the similarity of texture, and K_{dt} is a weighting ratio of the observed texture over the distance measurements. A larger K_{dt} indicates that the measurement model trusts the texture more than the distance measurements. K_{dt} is at least 1.

The similarity of distance measurements ξ_d is computed with a Mahalanobis distance function. The diagonal elements of the covariance matrix in the Mahalanobis distance function represent the uncertainty of each distance measurement. The uncertainty is proportional to the magnitude of the distance in most cases. However, because some distance measurements are unavailable due to the maximal measurable distance or the undetected boundary pixels in the image, the corresponding diagonal elements are set to a very large number. In addition, a particle is removed if the number of valid distance measurements is less than a predefined threshold.


(a) Actual observed texture.

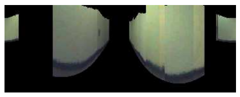

(b) Expected observed texture.

Fig. 5. An example of the similarity of texture.

The similarity ξ_t between the expected texture and the observed texture is computed by a pairwise pixel comparison as,

$$\xi_t = \frac{\#\ pixels\ in\ similar\ color}{\#\ total\ pixels}. \qquad (2)$$

A predefined threshold determines the maximal color difference for two pixels. Also, a normalized color space is adopted to alleviate the effect of illumination. Given a particle, the expected observed texture for the particle's pose can be projected from the 3D texture map. First, the relative horizontal distances to all neighbor boundary points are found. With the relative horizontal distance of a boundary point, the boundary pixels in the image plane can be found using the distance sensing function. The maximal observable heights of the vertical plane are calculated with the height sensing function. The texture on the vertical plane in the 3D texture map can then be projected to the corresponding pixels on the image plane as the expected observed texture. Fig. 5-(a) and (b) provide an example of an observed and expected texture in a panoramic view. The black pixels in Fig. 5(b) indicate the unavailable texture.

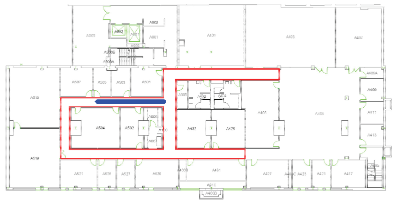
IV. Experiments

This section shows the performance of the localization approach with three experiments: pose tracking, global localization, and the kidnapped robot problem. Furthermore, the proposed approach is compared to distance-only MCL and SIFT-based MCL.

A. Pose Tracking and Global Localization

Pose tracking and global localization differ in whether the robot's initial pose in the map is given. With a known initial pose, the pose tracking tests if a localization approach can keep track of the robot's pose after the robot moves and accumulates error. On the other hand, global localization tests if the robot can eventually localize its pose in the map when the initial pose is unknown.

The experiments for pose tracking and global localization are conducted on the first floor and basement floor A of New Simon Hall (NSH) at Carnegie Mellon University. The experiments spanned the red regions shown in Fig. 6. The size of both regions is approximately 3556-by-1524 cm^2. The trajectory for the test is drawn as a blue line in Fig. 6(a). Odometry and 29 omnidirectional images are stored to test MCL in the texture 3D map that was created beforehand. K_{dt} is set as 2 for this experiment.


(a) Basement floor A.

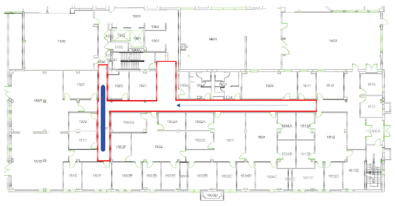
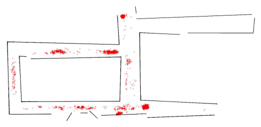

(b) First floor.

Fig. 6. Floor plan experiments


(a) After 5 resampling steps.

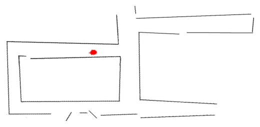
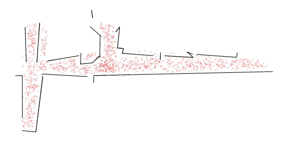

(b) After 14 resampling steps.

Fig. 7. Global Localization and pose tracking using MCL in the 3D texture map.

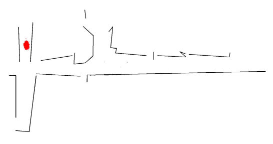
The results of global localization and pose tracking are shown in Fig. 7. In the beginning, 65000 particles are spread out on the first floor and floor A. After 10 resampling steps, 36245 particles remain and are located in the same hallway near the true pose. After 14 resampling steps, the remaining number of particles is 8409 and all of the particles converge to a local region around the real robot's pose. The local region is modeled as an ellipse with major axis of 106.7 cm and minor axis of 81.3 cm. The pose of the robot is continuously tracked until the robot stops.

B. The Kidnapped Robot Problem

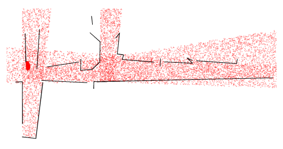
The test of the kidnapped robot problem evaluates the ability of a localization approach to recover from a failed localization procedure due to a noisy measurement or an unmodeled disturbance. Compared to global localization, the kidnapped robot problem is more difficult because the robot has to detect the instance that it is "kidnapped", and then it needs to relocalize itself in the environment. In our approach, a kidnapped robot is detected when the similarity of the measurements from all particles is low. This would not be the



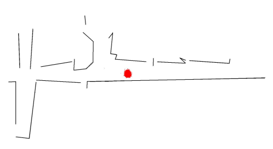
(a) After 1 resampling step.



(b) After 26 resampling steps.



(c) After 27 resampling steps - initialization of global localization.



(d) After 42 resampling steps.

Fig. 8. Solving the kidnapped robot problem with the proposed approach.

case normally because a high similarity would be expected from the true pose. After a kidnapped robot is detected, the particles are reinitialization. [6], [8] propose different methods to spread out the particles.

The trajectory for the test is shown as the two blue lines in Fig. 6(b). The robot moves along the trajectory drawn as the blue solid line and then is kidnapped and moved along the trajectory drawn as the blue dotted line. 65000 particles are spread out in the beginning on the first floor and floor A. After 21 resampling steps, about 12000 particles are used to track the robot's pose. The kidnapping occurs between the 26th resampling and the 27th resampling. When the robot kidnapping is detected, 12000 particles are used to track the predicted pose and 53000 particles are spread out in the environment randomly. The robot eventually finds the new pose after 42 resampling steps.

C. Performance Comparison

To evaluate our method, we compared our performance with a distance-only MCL implementation and a SIFT-based MCL implementation.

1) Comparison with Distance-only MCL: We compare distance-only MCL to the proposed approach in order to demonstrate the benefit of introducing the texture. The same distance sensing function and motion model described above are used in distance-only MCL. The weight of each particle is determined only by ξ_d.

Table I shows the comparison. The trajectory for the test and the omnidirectional images are the same as that used in Section IV-A. Not all of the particles in the distance-only MCL converge to the true pose after 29 resampling steps. The localization error of the proposed approach is 60.4 cm. Geometric similarity in the environment actually causes the

TABLE I

COMPARISON WITH DISTANCE-ONLY MCL

	Convergence Timestep	Error (cm)	Size of Particles (cm^2)
Dist.-Only MCL	Failed	Failed	Failed
Proposed Method	14	60.4	106.7×81.3

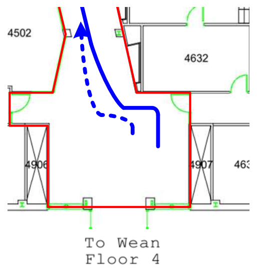


Fig. 9. Environment for our experiment on the 4th floor of NSH.

slow convergence of the distance-only MCL implementation. We attribute the faster convergence of the proposed approach to the valuable information that color and texture can provide for inferring a robot's pose in a known environment.

2) Comparison with SIFT-based MCL: SIFT-based MCL [3] is also compared to the proposed approach. The same motion model in Section III-A is adopted and the weight of each particle is computed as follows,

$$weight = \begin{cases} N \exp(-\frac{(d-\sigma)^2}{\tau^2}) & (d > \tau) \\ N & (d \leq \tau), \end{cases} \tag{3}$$

where N is the number of matched SIFT features between the current view and the omnidirectional image captured when originally mapping the space nearest to the particle; d is the distance between the particle and the nearest pose; τ and σ are both set as two times the minimal distance between the two poses where omnidirectional images are captured.

Experiments were performed on the 4th floor of NSH, as the red region shown in Fig. 9. The blue solid line represents the trajectory where the robot acquired omnidirectional images to originally build the map. Table II shows the localization results of the proposed approach and SIFT-based MCL with different σ and without σ when the robot moved in the trajectory drawn with a blue dotted line. The maximal displacement between these two trajectories is about 91.4cm.

SIFT-based MCL fails to localize the robot when σ is 13.5 cm. The reason for the failure is the nearly zero exponential

TABLE II

COMPARISON WITH SIFT-BASED MCL

	Error (cm)	Size of particles (cm^2)
SIFT-Based MCL(σ = 13.5)	Failed	Failed
SIFT-Based MCL(σ = 76.2)	204.7	239.6×102.9
SIFT-Based MCL(σ = 152.4)	150.4	311.5×99.6
SIFT-Based MCL(σ = ∞)	139.5	314×106.7
Proposed Method	100.6	178.5×117.3

TABLE III

COMPARISON WITH SIFT-BASED MCL WITH DISTANCE MEASUREMENT

	Convergence Timestep	Error (cm)	Size of Particles (cm^2)
Compared Method ($K_{ds} = 1$)	13	57.9	156.5×90.2
Compared Method ($K_{ds} = 2$)	9	54.8	139.2×81.3
Compared Method ($K_{ds} = 3$)	8	54.5	134.6×68.6
Proposed Method ($K_{dt} = 1$)	21	62.3	116.8×83.8
Proposed Method ($K_{dt} = 2$)	14	60.4	106.7×81.3
Proposed Method ($K_{dt} = 3$)	8	53.6	110.9×72.7

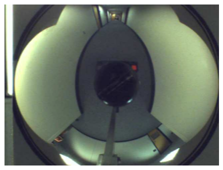
term in Eq. (3) due to a large d. When σ increases to 76.2, 152.4, or even infinite cm, the particles successfully converge to the real pose. Compared to the proposed approach which localizes the robot successfully with the error of 100.6 cm, the localization error of SIFT-based MCL under different σ is larger. The significance is that, unless tuned carefully, SIFT-based localization can fail when the path performed by the robot does not overlap well with the path that was taken when the images were originally acquired for building the map. Our method is not susceptible to this problem.

3) Comparison with SIFT-based MCL with Distance Measurements: In one final experiment, SIFT-based MCL is combined with distance measurements to compare with the texture-based method of the proposed approach. The weight of each particle is computed by $weight = \xi_d N^{K_{ds}}$, where K_{ds} is the weighting ratio of number for matched SIFT features over distance measurement.

The comparison is done in the same environment as the experiment in Section IV-A in order to evaluate the performance of SIFT-based MCL with distance measurements when compared to the proposed approach under different weight ratios. The localization result is summarized in Table III. When a high weight ratio is adopted, the proposed method is competitive with the SIFT-based MCL approach that includes distance measurements with respect to the time to convergence and localization accuracy. Additionally, both SIFT-based MCL and the proposed approach converge more quickly when the weight of each particle depends more on the number of matched SIFT features and the observed texture. However, the convergence speed cannot be further improved by giving a higher weight ratio because the robot is not close enough to distinct objects, such as a door, until the 8th omnidirectional image is captured. The 1st and the 8th omnidirectional images are shown in Fig. 10.

V. CONCLUSION AND FUTURE WORK

This paper proposes a KLD-based localization approach based on MCL in a 3D texture map. Both distance measurements and observed textures are used to decide the weight of each particle. The proposed approach converges faster than distance-only MCL approach and localizes the robot more precisely than the SIFT-based MCL approach. In addition, all of the hypotheses are better localized around the true pose than with SIFT-based MCL. Future work will focus on the localization approach in a map of 3D color point cloud and how to reduce the computation time.


(a) 1st Omnidirectional Image.

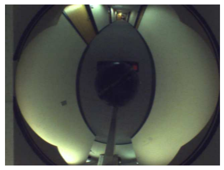

(b) 8th Omnidirectional Image.

Fig. 10. Two omnidirectional images during localization.

VI. ACKNOWLEDGEMENTS

Yu Fu gratefully acknowledges the grant by the National Science Council, Taiwan. (NSC98-2917-I-011-102)

REFERENCES

[1] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *International Journal of Robotic Research*, vol. 22, no. 12, pp. 985–1004, 2003.

[2] E. Megegatti, A. Pretto, A. Scarpa, and E. Pagello, "Omnidirectional vision scan matching for robot localization in dynamic environments," *IEEE Transactions on Robotics*, vol. 22, no. 3, pp. 523–535, 2006.

[3] H. Tamimi, H. Andreasson, A. Treptow, T. Duckett, and A. Zell, "Localization of mobile robots with omnidirectional vision using particle filter and iterative SIFT," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 758–765, 2006.

[4] M. Jogan and A. Leonardis, "Robust localization using an omnidirectional appearance-based subspace model of environment," *Robotics and Autonomous Systems*, vol. 45, no. 1, pp. 51–72, 2003.

[5] C. Siagian and L. Itti, "Biologically inspired mobile robot vision localization," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 861–873, 2009.

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

[7] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1322–1328.

[8] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro, "Image-based Monte Carlo localisation with omnidirectional images," *Robotics and Autonomous Systems*, vol. 48, pp. 17–30, 2004.

[9] D. Scaramuzze, A. Harati, and R. Siegward, "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2007, pp. 4164–4169.

[10] J. Mason, S. Ricco, and R. Parr, "Textured occupancy grids for monocular localization without features," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.

[11] S. Jo, Q. M. Shahab, Y.-M. Kwon, and S. C. Ahn, "Indoor modeling for interactive robot service," in *Proceedings of the SICE-ICASE International Joint Conference*, 2006, pp. 3531–3536.

[12] C. Plagemann, C. Stachniss, J. Hess, F. Endres, and N. Franklin, "A nonparametric learning approach to range sensing from omnidirectional vision," *Robotics and Autonomous Systems*, vol. 58, no. 6, pp. 762–772, 2010.

[13] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, 2nd ed., 1998.

[14] D. Scaramuzza, "Ocamcalib: Omnidirectional camera calibration toolbox for matlab," Retrieved from the World Wide Web: http://robotics.ethz.ch/∼scaramuzza/Davide_Scaramuzza_files/Research/OcamCalib_Tutorial.htm.