Self-localizaion of Mobile Robot Equipped with Omnidirectional Camera Using Image Matching and 3D-2D Edge Matching

Daisuke Ishizuka
Department of Mechanical Engineering
Shizuoka University
3-5-1 Johoku, Naka-ku, Hamamatsu-shi,
Shizuoka 432-8561, Japan
f0030003@ipc.shizuoka.ac.jp

Atsushi Yamashita
Department of Precision Engineering
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo 113-8656, Japan
yamashita@ieee.org

Ryosuke Kawanishi Toru Kaneko
Department of Mechanical Engineering
Shizuoka University
3-5-1 Johoku, Naka-ku, Hamamatsu-shi,
Shizuoka 432-8561, Japan
{f5945015,tmtkane}@ipc.shizuoka.ac.jp

Hajime Asama
Department of Precision Engineering
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo 113-8656, Japan
asama@robot.t.u-tokyo.ac.jp

Abstract

Self localization of mobile robots is necessary when they accomplish autonomous tasks in a given environment. In this paper, we propose a method for self localization of mobile robots equipped with an omnidirectional camera. The proposed method is composed of two phases. The first phase is for global localization. The robot estimates its position and orientation by matching an acquired omnidirectional image to arbitrary viewpoint images generated from a 3D environment model. The second phase is for local localization. Position and orientation are estimated by matching 2D edge points to 3D measurement data of lines. Experimental results showed the effectiveness of the proposed method.

1. Introduction

A mobile robot that works instead of a human has been developed recent years. For a mobile robot, self-localization is important. In this paper, we propose a self-localization method for a mobile robot.

As related works, self-localization methods using GPS [1], landmarks (RFID tags [2], etc.), normal camera [3], or are proposed. However, GPS is not available for indoor environments. A landmark-based navigation requires to set many landmarks properly in the environment. A normal camera-based navigation method is difficult to capture learning images in large area because a normal camera has a narrow view angle. In this paper, we propose a method which solves these problems.

Object color and structure information are effective for a mobile robot localization. An omnidirectional camera makes it possible to obtain the information of a surrounding environment efficiently [4–6]. Image-based self-localization using omnidirectional images has been proposed [7]. The method first captures omnidirectional images at regular intervals (learning images). Next, the robot estimates its position and orientation by matching learning images and the input image. However, the method cannot estimate its position robustly if the position is far from locations where learning images were acquired.

Therefore, our proposed method generates learning images from a 3D environment model (Fig. 1) [8] [9]. The 3D environment model has a lot of data of the surrounding environment such as color, 3D structure and so on. Therefore, a 3D environment model is effective to generate the learning images. These learning images provide a stable localization on the whole region of the map. As a preparation of the proposed self-localization, a 3D environment model is constructed by using an omnidirectional image sequence. To obtain learning images at the whole viewpoints in the map, arbitrary viewpoint images (Fig. 2(a)) are generated from the constructed model. These learning images make it possible to estimate the robot position and orientation robustly at the whole area of the map.

By the image-based localization, however, it is difficult to estimate precise robot position and orientation because

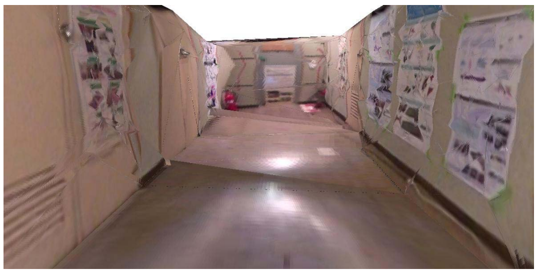
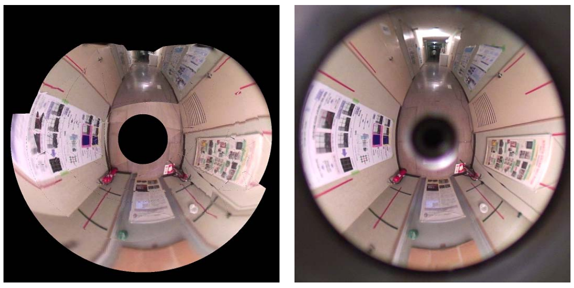


Figure 1. 3D environment model. The model has color and structure data of the surrounding environment.


(a) (b)
Figure 2. Omnidirectional images. These images have a 360-degree horizontal field of view. (a) Arbitrary viewpoint image example. The image is generated from a 3D environment model. (b) Real image. The image is captured by an omnidirectional camera.

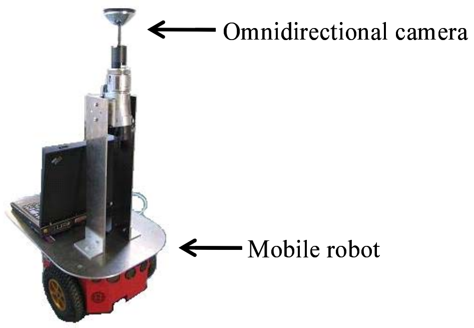
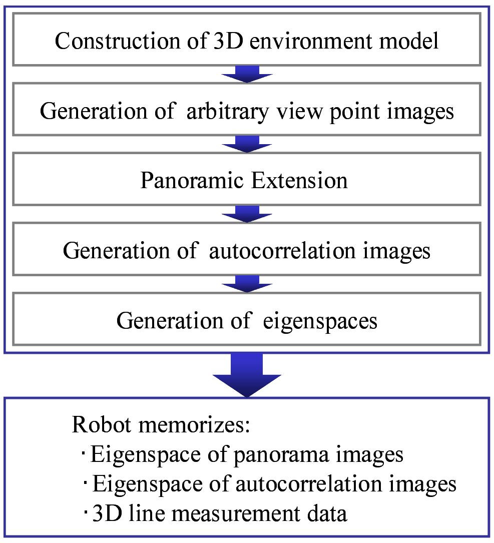


Figure 3. Mobile robot equiped with an omnidirectional camera.

discrete viewpoint images are created as learning images. Then, the proposed method utilizes line measurement data included in a constructed model. A precise robot location is estimated by matching 2D edge points of input image to 3D line measurements [10]. We propose a self-localization method which is composed of two phases: autocorrelation image matching (global self-localization) and 3D-2D edge matching (local self-localization).


Construction of 3D environment model
Generation of arbitrary view point images
Panoramic Extension
Generation of autocorrelation images
Generation of eigenspaces
Robot memorizes:
·Eigenspace of panorama images
·Eigenspace of autocorrelation images
·3D line measurement data
Figure 4. Learning process. The process is a preparation for navigation. The robot memorizes compressed images and the 3D environment model including 3D line mesurement data.

2. Process outline

In our proposed method, we use a mobile robot equipped with an omnidirectional camera as shown in Fig. 3. The camera attaches a hyperboloid mirror in front of the camera lens. We use the 3D environment model to generate learning images. The model is generated by our method based on the structure from motion (SFM) [8]. The model has color data, 3D structure data and 3D line measurement data.

The process of our method is shown in Fig. 4 and Fig. 5. Before self-localization, a preparation is needed. An omnidirectional image sequence is acquired during the robot locomotion. A three-dimensional environment model is constructed from the image sequence. A panoramic image at an arbitrary viewpoint in the 3D environment model is generated. Then, an autocorrelation image is calculated from the panoramic image. These two kinds of images are defined as learning images in the proposed method. Panoramic and autocorrelation images are compressed by using an eigenspace method. The robot memorizes these compressed images and the 3D environment model including 3D line measurement data.

The robot is navigated by using the learning images and the 3D line measurement data. In the global estimation phase, the robot position and orientation are estimated by matching an input image and the learning images. In the local estimation phase, the robot position and orientation are estimated precisely by matching 2D edge points to projected points of 3D line measurement data. Edge points are obtained from the input image by Canny edge detector [11].

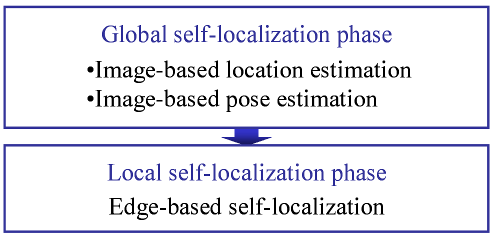

Global self-localization phase
•Image-based location estimation
•Image-based pose estimation
Local self-localization phase
Edge-based self-localization

Figure 5. Navigation process. The self-localization is composed of two phases: autocorrelation image matching (global self-localization) and 3D-2D edge matching (local self-localization).

3. Storing of learning images

3.1. Generation of arbitrary viewpoint images

An arbitrary viewpoint image generation method is mentioned below. An omnidirectional camera we use has a hyperboloid mirror in front of the conventional camera lens. A ray from an object point is reflected on the surface of the hyperboloid mirror. Then, the reflected ray heads to the center of the camera lens. All rays heading to the reflection point from the object point intersect at the focus of the hyperboloid mirror. Therefore, the focus of the hyperboloid mirror is regarded as the center of projection. A ray vector **r** = $[x, y, z]^T$ is defined as a unit vector heading to the reflection point on the mirror surface from the focus of the hyperboloid mirror as shown in Fig. 6. A ray vector **r** is calculated by the following equations from image coordinates $[u, v]^T$.

$$\mathbf{r} = \begin{bmatrix} su \\ sv \\ sf - c \end{bmatrix}, \tag{1}$$

$$s = \frac{a^2 \left(f\sqrt{a^2+b^2} + b\sqrt{u^2+v^2+f^2} \right)}{a^2 f^2 - b^2 \left(u^2+v^2\right)}, \tag{2}$$

where a, b and c are the hyperboloid parameters and f is the focal length of the lens, respectively.

The proposed method calculates a ray vector of image coordinates $[u, v]^T$ at the arbitrary viewpoint. The point at which the ray vector intersects with the 3D model is calculated. The pixel color at the image coordinates $[u, v]^T$ is determined as the same color of the cross-point. An arbitrary viewpoint image is generated by obtaining color information at all of the image coordinates.

3.2. Autocorrelation image

Even if omnidirectional images are captured at the same location, the appearance of the scenes will differ depending on the pose of the robot (Fig. 7). For this reason, these images can not match each other, even though they are taken at the same location. Therefore, we use autocorrelation images. The autocorrelation images are unique to

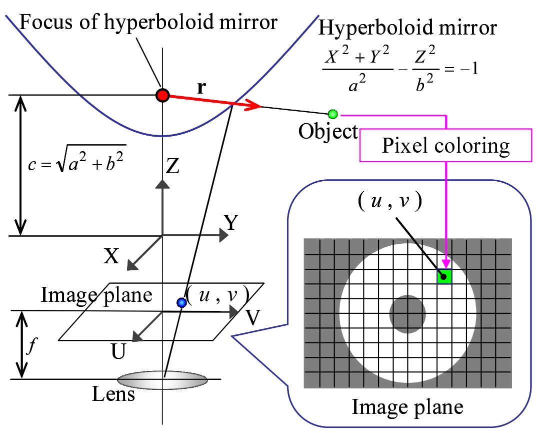


Figure 6. Calculation of ray vector and pixel coloring. A ray vector **r** is defined as a unit vector which starts from the focus of the hyperboloid mirror. The point at which the ray vector intersects with the 3D model is calculated. The pixel color at the image coordinates $[u, v]^T$ is determined as the same color of the cross-point.

the locations of capturing [7]. The autocorrelation image has a correlation value between a panoramic image (Fig. 8) and a horizontally-shifted panoramic image. Autocorrelation images having an extremal high level of similarity can be created from omnidirectional images taken at the same location (Fig. 9). Consequently, omnidirectional images taken at the same location can match each other by using autocorrelation images. The detailed explanation of autocorrelation image generation is mentioned below.

First, panoramic images are generated from omnidirectional images. The dimensions of the panoramic images are represented by $M \times N$ pixels. Autocorrelation function in the horizontal detection is calculated for each template consisting of a band-shaped region having size $M \times P$ $(1 \le P \le N)$ in the panoramic image. Specifically, if the n-th template $(n = 1, ..., N-P+1)$ from the top is expressed as $T_n(x, y)$ (where $x = 1, ..., M, y = 1, ..., P$), the autocorrelation function $r_n(k)$ $(1 \le k \le M)$ corresponding to $T_n(x, y)$ can be calculated as

$$r_n(k) = \frac{\sum_{x=1}^{M}\sum_{y=1}^{P}\left(T_n(x,y)-\overline{T}_n\right)\left(T_n(x+k,y)-\overline{T}_n\right)}{\sum_{x=1}^{M}\sum_{y=1}^{P}\left(T_n(x,y)-\overline{T}_n\right)^2}, \tag{3}$$

where $\overline{T}_n$ is the average of all pixel values for template $T_n(x, y)$. $r_n(k)$ computed using Eq. (3) indicates the normalized correlation value between template$T_n(x, y)$ and the value obtained by shifting this template by k pixels. The two-dimensional image expression of the $M \times (N-P+1)$ correlation values obtained by Eq. (3) for all $(N-P+1)$ templates is called an autocorrelation image. However, be-

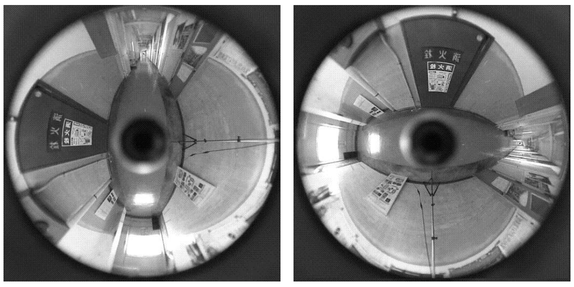

(a) (b)

Figure 7. Omnidirectional images captured at the same location. (a) An omnidirectional image captured at a location. (b) An omnidirectional image captured at the same location with camera rotation by 90 degrees.

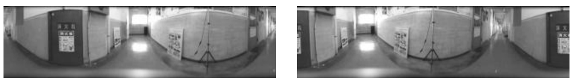
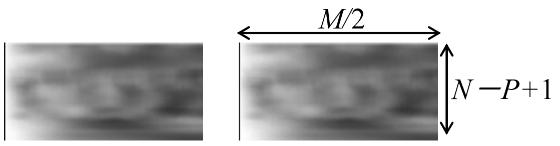

(a) (b)

Figure 8. Panoramic images captured at the same location. (a) A panoramic image captured at a location. (b) A panoramic image captured at the same location with camera rotation by 90 degrees.



(a) (b)

Figure 9. Autocorrelation images generated from the same location. (a) An autocorrelation image generated from a location. (b) An autocorrelation image generated from the same location with camera rotation by 90 degrees.

cause of the symmetry in the generated autocorrelation image, only half of its region actually needs to be considered, and an autocorrelation image having size $M/2\times(N-P+1)$ is generated.

3.3. Image storing by eigenspace method

In a case in which the navigation area is large, a lot of memory is needed. To reduce the size of memory, we compress image data by using the eigenspace method [12]. The eigenspace method is a linear feature extraction method that forms an eigenspace using the figure axis determined based on Karhunen-Loeve (KL) expansion, and compresses image data by projecting images onto the eigenspace. In the eigenspace, images are shown as points.

Consider an image set consisting of L images. Each image is expressed as an image vector $\mathbf{v}_j(j = 1, ..., L)$ that has pixel brightness values as its elements. In this case, a covariance matrix C corresponding to the image set can be determined as

$$\mathbf{C} = \frac{1}{L}\sum_{j=1}^{L}(\mathbf{v}_j - \overline{\mathbf{v}})(\mathbf{v}_j - \overline{\mathbf{v}})^T, \tag{4}$$

where $\overline{\mathbf{v}}$ is the average vector for $\mathbf{v}_j(j = 1, ..., L)$. The eigenspace corresponding to an image vector set $\mathbf{v}_j$ is formed using the eigenvector $\mathbf{u}_k(k = 1, ..., K)$ determined by solving the characteristic equation related to $\mathbf{C}$ described below as the orthogonal basic vector:

$$\mathbf{C}\mathbf{u}_k = \lambda_k\mathbf{u}_k. \tag{5}$$

When $a_{jk} = (\mathbf{v}_j - \overline{\mathbf{v}})^T\mathbf{u}_k$, the image vector $\mathbf{v}_j$ can be approximated as follows using K higher-ranking eigenvectors having large eigenvalues:

$$\mathbf{v}_j \simeq \overline{\mathbf{v}} + \sum_{k=1}^{K} a_{jk}\mathbf{u}_k. \tag{6}$$

Eq. (4) means that each image is transformed from a point in the original n-dimensional space into a point in the K-dimensional eigenspace having a lower dimension, thereby compressing the image data.

4. Self-localization

4.1. Global self-localization

4.1.1 Image-based location estimation

An image projected into the eigenspace is expressed as a point in the eigenspace. The Euclidean distance between points in the eigenspace is used as the measure of image similarity in the eigenspace. The Euclidean distances between the input image and learning images are calculated. The location where the input image is acquired is determined as the learning location having the minimal Euclidean distance.

4.1.2 Image-based pose estimation

The robot has panoramic images compressed in the eigenspace. To estimate robot orientation, a learning panoramic image is shifted little by little. The correlation values between the input panoramic image and the shifted images are calculated by the following equation.

$$R(k) = \frac{\sum_{j=1}^{N}\sum_{i=1}^{M}\left\{\left(I(i,j) - \overline{I}\right)\left(T(i+k,j) - \overline{T}\right)\right\}}{\sqrt{\sum_{j=1}^{N}\sum_{i=1}^{M}\left(I(i,j) - \overline{I}\right)^2 \sum_{j=1}^{N}\sum_{i=1}^{M}\left(T(i+k,j) - \overline{T}\right)^2}}, \tag{7}$$

where $\overline{I}_n$ is the average for all pixel brightness values $I_n(x, y)$ of learning images, and $\overline{T}_n$ is the average for all

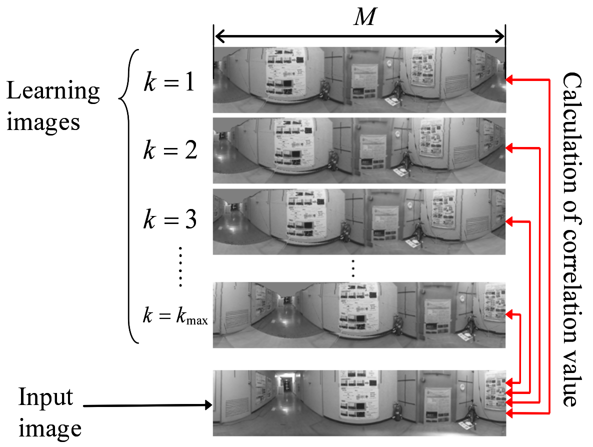


Figure 10. Pose estimation method. A learning panoramic image is shifted little by little. The correlation values between the input panoramic image and the shifted images are calculated.

pixel brightness values $T_n(x + k, y)$ of the input image. When a correlation value between the learning image and an input image become the maximum, the robot orientation t[rad] is calculated from the shift amount of the learning image k_{max} by the following equation (Fig. 10).

$$t = \frac{k_{max}}{M} \times 2\pi. \quad (8)$$

4.2. Local self-localization

For precise self-localization, the proposed method optimizes the robot position and orientation by using 3D-2D edge matching. An edge-based localization is robust for brightness change in the environment [13]. For 3D-2D matching based localization, the proposed method uses 2D Canny edge points and measurement data of 3D lines (Fig. 11).

The robot position and orientation estimated by an image-based self-localization are used for initial values in the following process.

4.2.1 3D edge point projection to omnidirectional image

To match between 2D Canny edge points (Fig. 12(a)) and measurement data of 3D lines, 3D edge points are projected to 2D image coordinates $[u, v]^T$ (Fig. 12(b)) by using ray vector **r** heading to the 3D edge points from the focal point of the hyperboloid mirror:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} s's''x \\ s's''y \end{bmatrix}, \quad (9)$$

$$s' = \frac{cz + b\sqrt{x^2 + y^2 + z^2}}{\left(\frac{b}{a}\right)^2 (x^2 + y^2) - z^2}, \quad (10)$$

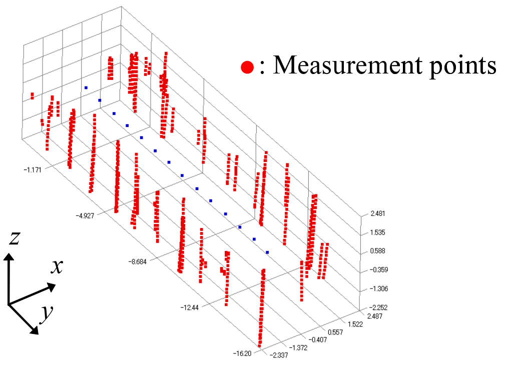


Figure 11. 3D edge points. Red points are 3D edge points. The edge points data are obteined from the 3D environment model.

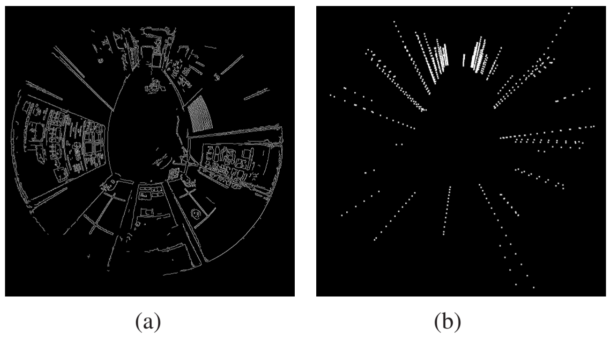


Figure 12. Edge points. (a) The edge points are obtained from an input image by Canny edge detector. (b) The edge points are projected from the 3D environment model.

$$s'' = \frac{f}{s'z + 2c}. \quad (11)$$

4.2.2 3D-2D edge matching

The robot position and orientation are optimized by matching Canny edge points and projected edge points. Here, the number of the 3D edge points is N, and these projected image coordinates are (a_{xi}, a_{yi}). Image coordinates of the Canny edge point which has the minimum distance to projected image coordinates (a_{xi}, a_{yi}) is (b_{xi}, b_{yi}). A sum of distances $D(x, y, \theta)$ between (a_{xi}, a_{yi}) and (b_{xi}, b_{yi}) is calculated by the following equation.

$$D(x, y, \theta) = \frac{1}{N}\sum_{i}^{N}\left((a_{xi} - b_{xi})^2 + (a_{yi} - b_{yi})^2\right). \quad (12)$$

The position and orientation which have the minimum value of $D(x, y, \theta)$ are determined as the robot position and orientation.

5. Experiments

5.1. Image memory self-loclization

In this section, we evaluate the accuracy of the image memory-based self-localization at an indoor environment (corridor). The robot moves along the center of the corridor (Fig. 13: dotted line). In advance, a 3D environment model is constructed from an omnidirectional image sequence. 50 learning images are generated from the 3D environment model at 0.3m intervals as shown in Fig. 13. 10 input images are captured along the same path where the robot moved beforehand (Fig. 14: red arrows). 20 input images are captured along the other path (Fig. 14: purple, green and blue arrows). The robot performed self-localization by matching the input images and learning images. We evaluate the average of self-localization correct answers. Here, the correct answer is a cace in which the robot chooses the learning image that is the nearest to the location of the robot. Then the robot estimates its orientation.

The robot chose correct answers at 29 of the 30 locations. The robot can estimate its position and orientation at the location the robot has never moved to. The result of the self-localization is shown in Fig. 15 (successful) and Fig. 16 (failed). These figures are the top views of the corridor and show the correlation values between the input image and the learning image. In these figure, the location at which the correlation value is the highest (in pink frame) is regarded as the location of the robot. If the location of the robot and the location of input image capture are same, the self-localization is performed correctly.

In Fig. 15, the estimating location of the robot and the location of input image capture are same, so the self-localization is successful. In Fig. 16, the estimating lolcation of the robot and the location of input image capture are not same, so the self-localization is failed. The cause of the failure is that there are a lot of locations at which correlation value between an input image and a learning image is high in an environment whose change of pixel brightness value is low. The average error of the image-based location estimation is 47.6mm. The processing time of location estimation is 0.53 seconds. The average error of pose estimation is 2.1 degrees. Processing time of pose estimation is 0.09 seconds.

5.2. 3D-2D edge matching based self-localization

In this section, we evaluated the accuracy of the 3D-2D edge matching self-localization. The initial location is given by image-based self-localization. 400 images projected by 3D edge points are generated at 1cm intervals around the initial location. At each location, 11 images of 3D edge point projections are generated at 1 degree intervals around the initial orientation, and a total of 4400 images which are projections of 3D edge points are generated. As the evalua-

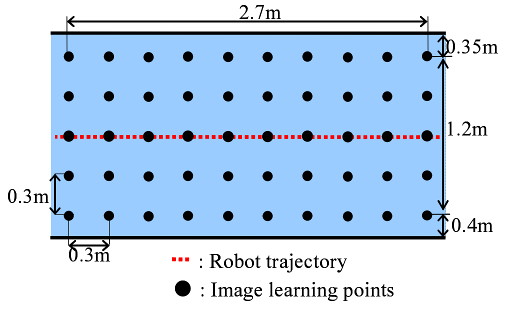


Figure 13. Image learning points. Learning images are generated at black points. The robot moves along the red dotted line to caputure omnidirectional image sequence in advance.

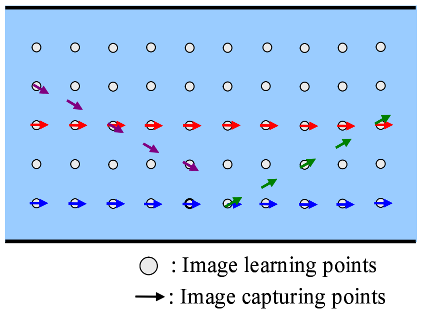


Figure 14. Input images capturing points. Input images are captured at the location of arrows. 10 input images are captured along the same path where the robot moved beforehand (red arrows). 20 input images are captured along the other path (purple, green and blue arrows). White points are image learning points.

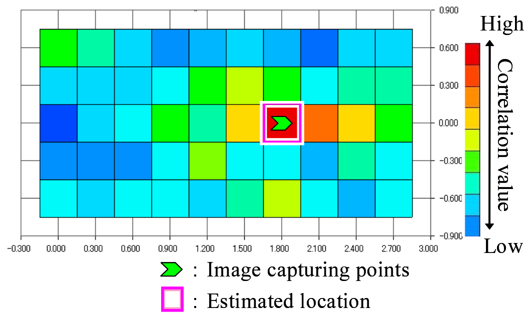


Figure 15. Result of self-localization (successful). The color shows correlation values between the input image and learning images. Pink frame is the estimation location at which the correlation value is the highest. Green arrow is the input image capturing point.

tion value of self-localization, we use $D(x, y, \theta)$ (Eq.(12)).

When image-based self-localization and edge-based self-localization are performed, $D(x, y, \theta) = 46949.4$. On the other hand, when only image-based self-localization is performed, $D(x, y, \theta) = 57828.9$. This shows that edge-

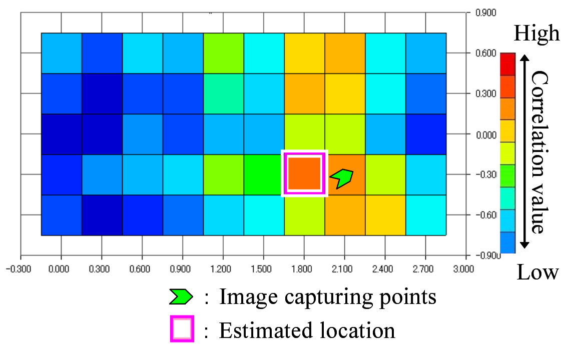


Figure 16. Result of self-localization (failed). The color shows correlation values between the input image and learning images. Pink frame is the estimation location at which the correlation value is the highest. Green arrow is the input image capturing point.

Table 1. Experiment result.

	Image-based	Image-based and edge-based
Average error of location estimation	47.6mm	11.4mm
Average error of pose estimation	2.1deg	0.6deg
Processing time	0.62sec	4.85sec

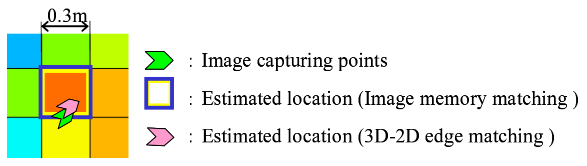

0.3m
: Image capturing points
: Estimated location (Image memory matching)
: Estimated location (3D-2D edge matching)

Figure 17. Comparison of image memory matching with 3D-2D edge matching.

based self-localization is effective. In Fig. 17, we can recognize that edge-based self-locazilation is more accurate than image-based self-localization. An experiment result is shown in Table. 1. The average error of the edge-based location estimation is 11.4mm. The average error of the 3D-2D edge matching based pose estimation is 0.6 degrees. Processing time of the 3D-2D edge matching based self-localization is 4.23 seconds. Shortening the processing time of edge-based self-localization is possible by cutting the number of matching patterns.

6. Conclusions

This paper proposed a self-localization method using a 3D environment model for a mobile robot equipped with the omnidirectional camera. By using arbitrary viewpoint images generated from a 3D environment model as learning images, the robot can estimate its poistion and orientation at the location where the robot did not move beforehand. Also, by adding 3D-2D edge matching, the robot can estimate its position and orientation accurately. Experimental result showed the effectiveness of the method. Future works are shortening of the processing time of 3D-2D edge matching based self-localization, performing of a long distance navigation and establishing of a navigation method at dynamic environment.

Acknowledgements

This work was in part supported by MEXT KAKENHI, Grant-in-Aid for Young Scientist (A), 22680017.

References

[1] S-H. Kim, C-W. Roh, S-C. Kang and M-Y. Park: "Outdoor navigation of a mobile robot using differential GPS and curb detection", *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pp.3414–3419, 2007.

[2] V. Kulyukin, C. Gharpure, J. Nicholson and S. Pavithran: "RFID in robot-assisted indoor navigation for tha visually impaired", *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.1979–1984, 2004.

[3] M. Cummins and P. Newman: "FAB-MAP: Probabilistic localization and mapping in the space of appearance", *International Journal of Robotics Research*, Vol. 27, No. 6, pp.647–665, 2008.

[4] A. C. Murillo, J. J. Guerrero and C. Sagues: "SURF features for efficient robot localization with omnidirectional images", *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pp.23–30, 2007.

[5] S. Ramalingam, S. Bouaziz, P. Sturm and M. Brand: "Geolocalization using skylines from omni-images", *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops*, pp.3901–3907, 2009.

[6] G. L. Mariottini and D. Prattichizzo: "Uncalibrated video compass for mobile robots from paracatadioptric line images", *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.226–231, 2007.

[7] H. Iwasa, N. Aihara, N. Yokoya and H. Takemura: "Memory-based self-localization using omnidirectional images", *Systems and Computers in Japan*, Vol. 34, No. 5, pp.56–68, 2003.

[8] R. Kawanishi, A. Yamashita and T. Kaneko: "Estimation of camera motion with feature flow model for 3D environment modeling by using omni-directional camera", *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp.3089–3094, 2009.

[9] B. Micusik, D. Martinec and T. Pajdla: "3D metric reconstruction from uncalibrated omnidirectional camera", *Proceedings of the 2004 Asian Conference on Computer Vision*, pp.545–550, 2004.

[10] E. Marchand, P. Bouthemy, F. Chaumette and V. Moreau: "Robust real-time visual tracking using a 2D-3D model-based approach", *Proceedings of the 7th IEEE International Conference on Computer Vision*, Vol. 1, pp.262–268, 1999.

[11] J. F. Canny: "A computional approach to edge detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 6, pp.679–698, 1986.

[12] M. A. Turk and A. P. Pentland: "Face recognition using eigenface", *Proceedings of the 1991 IEEE Conference on Computer Vision and Pattern Recognition*, pp.586–591, 1991.

[13] M. Tomono: "Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm", *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pp.4306–4311, 2009.