IECON2015-Yokohama
November 9-12, 2015

Statistical Localization Exploiting Convolutional Neural Network for an Autonomous Vehicle

Satoshi Ishibushi, Akira Taniguchi, Toshiaki Takano, Yoshinobu Hagiwara and Tadahiro Taniguchi
Ritsumeikan University
1-1-1 Noji-higashi, Kusatsu,
Shiga 525-8577, Japan
E-mail: ishibushi@em.ci.ritsumei.ac.jp

***Abstract*—In this paper, we propose a self-localization method that exploits object recognition results by using convolutional neural networks (CNNs) for autonomous vehicles. Monte-Carlo localization (MCL) is one of the most popular localization methods that use odometry and distance sensor data for determining vehicle position. Some errors are often observed in the localization tasks and MCL often suffers from global positional errors. A global positional error means that particles representing a vehicle's position are distributed in the form of a multimodal distribution, i.e., the distribution has several peaks. To overcome this problem, we propose a method in which an autonomous vehicle employs object recognition results, obtained using CNNs, as the measurement data with a Bag-of-Features representation in an integrative manner. The semantic information found in the recognition results obtained using the CNN reduces the global errors in localization. The experimental results show that the proposed method can converge the distribution of the vehicle positions and particle orientations and reduce the global positional errors.**

I. Introduction

In the research field of autonomous vehicles and mobile robots, it is important for a robot to estimate self-location in order to move autonomously in its environment. To realize the self-localization of mobile robots, odometry data, environmental maps, and the observed data from sensors, i.e., range information from a robot for an obstacle, are usually used. Simultaneous localization and mapping (SLAM) is a statistical method that enables a robot to estimate its pose and builds a map of the environment at the same time [1]. In general, SLAM expresses maps as occupancy grid-based and feature-based maps. An occupancy grid map is built by employing range data obtained from the robot's sensors.

Monte Carlo localization (MCL) [2] is one of the most popular self-localization methods, which usually uses odometry and distance sensor data. These data often include noise such as slipping wheels and measurement errors. MCL can correct these errors locally on the basis of Bayesian filtering and sampling importance resampling procedures. However, global errors are often not preserved without correction, and the posterior distribution of the estimation results is multimodal. The objective of this study is to enable a robot to reduce global errors in MCL on the basis of object recognition results, which are obtained from a convolutional neural network (CNN), which is one of the deep learning methods.

In the research field of image recognition, CNN has been attracting attention as a state-of-the-art method for images and object recognition [3]–[7]. CNN consists of several convolutional layers and pooling layers. However, there have been few studies that integrate the self-localization methods and CNN. In this study, we propose a method by using a robot that statistically exploits odometry and the distance sensor data and object recognition results of CNNs for self-localization. This section describes previous studies for the self-localization method. Karlsson et al. have proposed visual SLAM (vSLAM), which uses the local features of the observed images as landmarks, and simultaneously estimates the self-location and position of landmarks [8]. Further, Ahn et al. have proposed a SLAM method, which is based on visual object recognition [9]. Their method uses objects that are registered in the database as landmarks. Thus, the method enables a robot to detect more robust landmarks than by using local features such as a scale-invariant feature transform (SIFT), speed-up robust features (SURF), and oriented fast and rotated BRIEF (ORB). These studies [8], [9] are similar in the way they exploit the object recognition results for the self-localization method. Hu et al. have proposed RGB-D SLAM, which builds a three-dimensional (3D) map by using the range date and RGB information [10]. RGB-D SLAM employs the RGB information that is obtained from the camera so that the method is allowed to make up for the extent and accuracy constraints of the range sensor. Hagiwara et al. have proposed view-based navigation methods that are robust against the changes in scenery for mobile robotics tasks [11], [12]. These methods realized the robust self-localization of a robot by using image matching algorithms based on SURF and the approximate plane of brightness.

Researchers [8], [9] have proposed a method that estimates the positions of landmarks and self-location. The number of landmarks is different for each place, and positional errors are larger in places where the robot cannot detect landmarks. Further, the robot often estimates different positions on the basis of the differences in vision. In contrast, in our research, a robot learns what image feature is obtained at each place by using the Bag-of-Features representation, which can be generated by a CNN. Therefore, the robot is allowed to estimate what feature is detected in each area of the environment. Furthermore, we assume that the robot can learn an invariant

978-1-4799-1762-4/15/$31.00 ©2015 IEEE

feature with respect to a few declinations in the robot position and the difference in view of the features extracted from images by using the estimation features for each place. RGB-D SLAM [10] builds a 3D map, and a robot can estimate self-location by using the RGB information from the map. Therefore, the robot can correct errors caused by odometry or distance sensor data. However, it needs a large memory to store a 3D map, and this leads to heavy computational costs. The view-based navigation methods [11], [12] can estimate the self-location of a robot with high accuracy. However, in the case of a long distance path, the cost of memorizing and matching images at each position increases, which is a problem. In this study, we propose a self-localization method that uses the object recognition results generated by a CNN as the features of each place. Since the robot uses the features of a place instead of the RGB information obtained from raw images, it needs less storage capacity and has a lower computational cost than the conventional approach.

To develop the proposed method, we extended Taniguchi et al.'s self localizable method, which combines MCL and semiotic information [13]. They statistically exploited words representing the name of the place for self-localization. On the other hand, we statistically exploited the features extracted from an image. For self-localization of a mobile robot, exploiting a CNN that generates positive results is one of the contributions of this research.

II. Proposed method

A. Proposed model for localization

In this paper, we propose a self-localization method that integrates the graphical model of MCL [2] and the landscape information, i.e., information about any particular object that can be seen in a particular place. In this section, we describe the proposed method. Fig. 1 shows that a graphical model for the proposed self-localization method. The individual elements of the graphical model can be defined as follows: The self-position of the robot at time t is denoted as $\boldsymbol{x_t}$. In this paper, the self-position of a robot is defined as follows:

$$\boldsymbol{x_t} = (x_t, y_t, \sin\theta_t, \cos\theta_t). \tag{1}$$

x_t and y_t denote the coordinates of a plane on the floor. θ_t represents the angle of the robot. The angle θ_t indicates that the angle to the x axis is 0° and the angle to the y axis is 90°. The odometry value and the observation value of the robot at time t are denoted as u_t and z_t, respectively. In this study, we define a "region" as an area considering the positions and similarity of features extracted from an image observed by a mobile robot. An index of a "region" is denoted as C_t and is defined as follows: $C_t \in \boldsymbol{C} = \{1, 2, \cdots, L\}$. The number of "regions" is denoted as L. Furthermore, the feature vector of the image observed at time t is denoted as $f_t = (f_t^1, f_t^2, \cdots, f_t^I)^{\mathrm{T}}$. The number of elements of the feature vector is denoted as I. The probability representing the self-position of a robot $\boldsymbol{x_t}$ is obtained as follows:

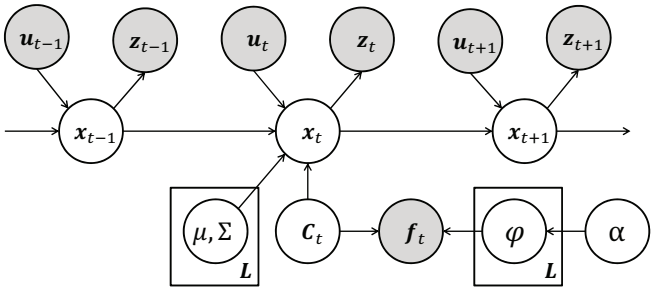

Fig. 1. Graphical model of the proposed method

$$\begin{aligned} p(\boldsymbol{x_{0:t}}|z_{1:t}, u_{1:t}, f_{1:t}) &\propto p(z_t|\boldsymbol{x_t})p(f_t|\boldsymbol{x_t})p(\boldsymbol{x_t}|\boldsymbol{x_{t-1}}, u_t) \\ &\quad \times p(\boldsymbol{x_{0:t-1}}|z_{1:t-1}, f_{1:t-1}, u_{1:t-1}). \end{aligned} \tag{2}$$

Then, the posterior distribution $p(f_t|\boldsymbol{x_t})$ representing feature vector f_t observed at the self-position of a robot $\boldsymbol{x_t}$ is obtained as follows:

$$p(f_t|\boldsymbol{x_t}) = \sum_{C_t} p(f_t|C_t)p(C_t|\boldsymbol{x_t}) \tag{3}$$

$$\propto \sum_{C_t} p(f_t|C_t, \varphi)p(\boldsymbol{x_t}|C_t, \mu, \Sigma)p(C_t) \tag{4}$$

$$= \sum_{C_t} p(f_t|\varphi_{C_t})p(\boldsymbol{x_t}|\mu_{C_t}, \Sigma_{C_t})p(C_t). \tag{5}$$

The Gaussian distribution that has mean vector μ and covariance matrix Σ denotes a position distribution of $\boldsymbol{x_t}$, and the multinomial distribution φ denotes a distribution representing the feature of the image. Equation (4) follows the Bayes rule. The hyperparameter for a Dirichlet prior distribution is $\boldsymbol{\alpha}$. The respective Gaussian distributions and multinomial distributions exist depending on the number of "regions." The mean vector and the covariance matrix of the Gaussian distribution of index C_t and the multinomial distribution of index C_t are denoted as μ_{C_t}, Σ_{C_t} and φ_{C_t}, respectively. Probability $p(f_t|\varphi_{C_t})$ of a feature vector observed in each "region" is obtained by

$$p(f_t|\varphi_{C_t}) = \mathrm{Mult}(f_t|\varphi_{C_t}, K) \tag{6}$$

where the parameter K is restricted as follows: $K = \sum_{i=1}^{I} f_t^i$. Furthermore, the probability representing the self-position of a robot in each "region" is obtained as follows:

$$\begin{aligned} p(\boldsymbol{x_t}|\mu_{C_t}, \Sigma_{C_t}) &= \frac{1}{\sqrt{2\pi}^m |\Sigma_{C_t}|^{1/2}} \\ &\quad \times \exp\left\{-\frac{1}{2}(\boldsymbol{x_t} - \mu_{C_t})^T {\Sigma_{C_t}}^{-1}(\boldsymbol{x_t} - \mu_{C_t})\right\}. \end{aligned} \tag{7}$$

The dimension of $\boldsymbol{x_t}$ is denoted as m. The prior probability of C_t assumes the uniform distribution as follows: $p(C_t) = \frac{1}{L}$.

B. Statistical exploitation of object recognition results obtained using CNN

In this paper, a feature vector f_t is obtained by using the object recognition results generated by a CNN as the categorization results. Features that the CNN extracts are invariant to image translation, rotation, and scale. Fig. 2 shows the overall architecture of a CNN. The fully connected layers are connected following the topmost pooling layer, and the CNN can obtain probabilities $P(O_i)$ representing that the input images are categorized into object class O_i by using a softmax function. A softmax function is defined as follows:

$$p(O_i) = \frac{e^{h_i}}{\sum_{k=1}^{I} e^{h_k}}. \tag{8}$$

The input values to the output layers are denoted as $h_i (i = 1, \cdots, I)$. The number of the object class is denoted as I. In this study, elements of the feature vector f_t are determined by using the categorization result. Then, the value s of the power of ten as well as probability $P(O_i)$ are employed as elements of the feature vector by using the Bag-of-Features representation. In other words, the respective elements of the feature vector ${f_t}^i$ are obtained as follows:

$${f_t}^i = \lfloor p(O_i) \times 10^s \rfloor. \tag{9}$$

For example, the value s is set as $s = 2$ and the probability $P(O_i)$ is obtained as $P(O_i) = 0.23$; then, the feature vector's element ${f_t}^i$ is regarded as ${f_t}^i = 23$. Fig. 3 shows an example of the object categorization result generated by the CNN. Each word in Fig. 3 denotes a label of the object class. Some classes are labeled with more than one object name. For example, it regards "mailbag" and "postbag" as the same object, and the label of the object includes both these terms.

Furthermore, numerical values that are shown on the right of each label denote the probability of an input image to be categorized into an object class. Fig. 3 shows the object classes of the top five probabilities beginning at the top. The topmost object class is "turnstile," and the corresponding probability is 0.0691.

In fact, the image shown in Fig. 3 is an image of "air cleaner." In contrast, in the categorization result, the image is categorized into the class label of another object class such as "turnstile" and "mailbag" with a high probability. However, in cases where results like this are often obtained, a robot can learn that the feature vector with a high value of the element of "turnstile" and "mailbag" can be observed in the "region." Therefore, in cases where the robot observes a feature vector with a high value of the elements of "turnstile" and "mailbag," the robot can estimate the self-location near the air cleaner with a high probability.

The parameter of the multinomial distribution φ_l is obtained by sampling from the Dirichlet posterior distribution that is derived from the Dirichlet prior distribution and the observation value as follows:

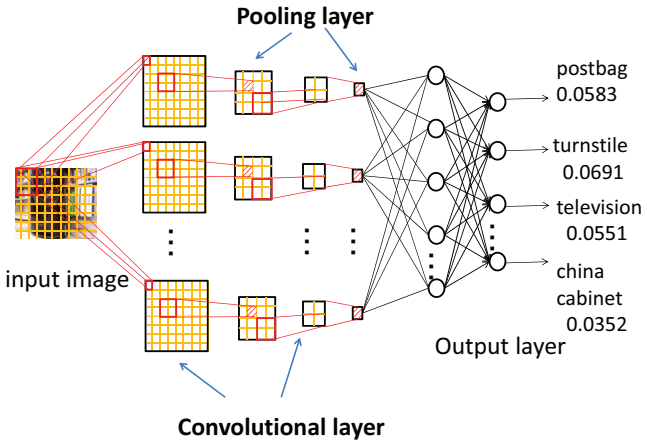

Fig. 2. Overall architecture of the CNN

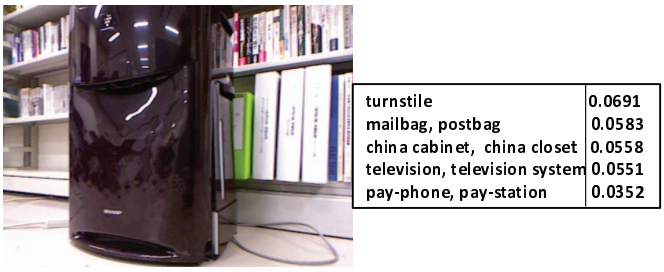

Fig. 3. Example of an object categorization result obtained using a CNN

$$\begin{aligned} \boldsymbol{\varphi}_l &\sim \mathrm{Dir}(\boldsymbol{\varphi}_l | \boldsymbol{\alpha} + \boldsymbol{m}) \\ &= \frac{\Gamma(\alpha_0 + K)}{\Gamma(\alpha_1 + m_1) \cdots \Gamma(\alpha_I + m_I)} \prod_{i=1}^{I} \varphi_{l,i}^{\alpha_i + m_i - 1}. \end{aligned} \tag{10}$$

The values of each element of the feature vector are denoted as $\boldsymbol{m} = (m_1, m_2, \cdots, m_I)^{\mathrm{T}}$. The hyperparameters of the Dirichlet distribution are denoted as $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_I)^{\mathrm{T}}$.

III. Experiment

A. Experimental conditions

1) Tools used: In this paper, we used Turtlebot 2[1]as the experimental robot. Turtlebot 2 is an open robotics platform that is implemented by using a robot operation system (ROS)[2]. We controlled Turtlebot 2 by DUAL SHOCK 3[3]that is a wireless controller. We conducted the experiment in the Emergent Systems Laboratory of Ritsumeikan University. In this study, we used Caffe [14], which is a framework for CNN. We used

[1]http://kobuki.yujinrobot.com/home-en/about/reference-platforms/turtlebot-2/

[2]http://wiki.ros.org/

[3]http://www.jp.playstation.com/ps3/peripheral/cechzc2j.html

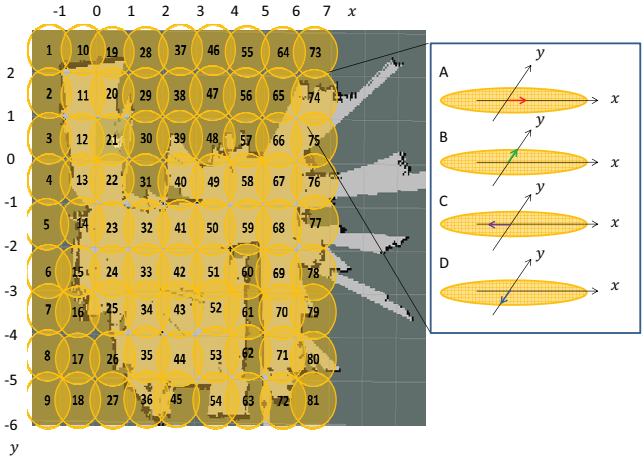

Fig. 4. Map of Gaussian distributions corresponding to the "regions"

the parameter of the CNN trained by using ImageNet Large Scale Visual Recognition Challenge 2013[4] as the dataset.

2) Region determination: In this study, we assumed that similar features are observed at close positions, and a "region" is determined by the closeness of positions. Each number in Fig. 4 shows the index of each "region," and the center of the Gaussian distribution related to the coordinates x, y is located in the middle of a grid. The mean of the Gaussian distribution at index 1 was set as $x_1 = -1.5, y_1 = 2.5$ in Fig. 4. x_n, y_n denotes the mean of the Gaussian distribution at index n and was obtained by using the following equation:

$$(x_n, y_n) = (x_1 + \left\lfloor \frac{n-1}{9} \right\rfloor, y_1 - n + 9 \left\lfloor \frac{n-1}{9} \right\rfloor + 1). \quad (11)$$

In Fig. 4, the angles are labeled as A, B, C, and D. These angles can be defined as $(\sin\theta = 0, \cos\theta = 1)$, $(\sin\theta = 1, \cos\theta = 0)$, $(\sin\theta = 0, \cos\theta = -1)$, and $(\sin\theta = -1, \cos\theta = 0)$, respectively. Furthermore, the covariance of every Gaussian distribution was set as follows:

$$\Sigma_l = \begin{pmatrix} \sigma_{xx} & \sigma_{yx} & \sigma_{(\cos\theta)x} & \sigma_{(\sin\theta)x} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{(\cos\theta)y} & \sigma_{(\sin\theta)y} \\ \sigma_{x(\cos\theta)} & \sigma_{y(\cos\theta)} & \sigma_{(\cos\theta)(\cos\theta)} & \sigma_{(\sin\theta)(\cos\theta)} \\ \sigma_{x(\sin\theta)} & \sigma_{y(\sin\theta)} & \sigma_{(\cos\theta)(\sin\theta)} & \sigma_{(\sin\theta)(\sin\theta)} \end{pmatrix}$$
$$= \mathrm{diag}(0.5, 0.5, 0.7, 0.7)$$

In this paper, the number of "regions" L is denoted as 324. In 90 "regions," the robot had observed some images. It observed 3 to 32 images in each "region."

3) Parameter setting: This section describes the parameters. The number of particles of MCL was set as 1000. The Dirichlet hyperparameter was set as 12. Furthermore, the parameter s in equation (9) was set as 2.

[4]http://www.image-net.org/challenges/LSVRC/2013/

B. Procedure for learning parameters of a multinomial distribution

This section describes the procedure for learning the parameters of the multinomial distribution of each "region."

1) **Building map by SLAM**
   The robot built a map by using SLAM.
2) **Estimating self-position and storing observed images**
   We made the robot estimate the self-position with the map, and the robot stored the observed image and the estimated position.
3) **Allocating images to a "region"**
   A robot allocated the stored images to each "region" by using an approximate expression as follows:
$$C_t = \arg\max_{C_t} p(\boldsymbol{x_t}|\mu_{C_t}, \Sigma_{C_t}). \quad (12)$$
4) **Categorizing images by using the CNN**
   Images in every "region" are categorized into an object class by the CNN.
5) **Using Bag-of-Features representation and learning the parameters of the multinomial distribution**
   The robot counted the values of the elements of the feature vectors by using the Bag-of-features representation, and learned the parameter φ of the multinomial distribution by using the results of the count.

In this study, we compared the following four methods for confirming the effectiveness of employing the object recognition results obtained by using the CNN.

- **MCL + CNN + Bag-of-Features (Proposed method)**
  This is the proposed method, which uses the object recognition results obtained by using the CNN. Then, the number of object classes I was set to 1000.
- **MCL + SIFT + Bag-of-Features**
  This method extracted SIFT from every observed image and clustered these features by using the K-means method. Then, we set the number of classes I as 200. In each "region," it counted the number of classes to which the extracted features from the observed image were allocated in the Bag-of-Feature representation. Each count of the classes was denoted as an element of the feature vector. The robot learned the parameter φ of the multinomial distribution on the basis of the count result for each "region." The robot extracted SIFT from an observed image and counted the number of classes allocated to each class. It estimated self-location by using the learning and count results.
- **MCL + RGB + Bag-of-Features**
  First, this method compressed the observed images to the size of 10×10 pixels and extracted the RGB color code of 256 tones from every compressed image. It clustered these color codes by using the K-means method. Then, we set the number of classes L as 200. In each "region," it counted the number of classes to which the extracted color codes from the observed image were allocated by using the Bag-of-Feature representation. Each count of the classes is denoted as an element of the feature vector.

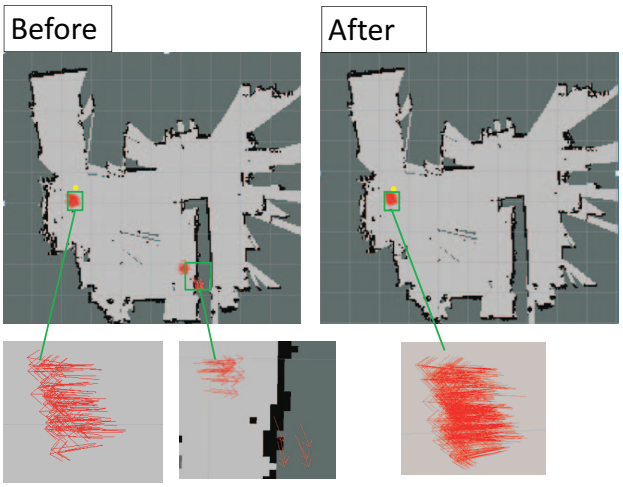

Fig. 5. Condition of particles, left:before reflecting an object categorization result, right: after reflecting an object categorization result. The yellow point in the figure denotes the actual position of the robot.

The robot learned the parameter φ of the multinomial distribution on the basis of the count result for each "region." The robot compressed an observed image to the size of 10×10 pixels and extracted the color code from every compressed image. It estimated self-location by using the learning and count results.

- **MCL (Baseline method)**
  The robot estimated self-location by MCL, which used only odometry and distance sensor data.

C. Estimated position and particle orientation

We qualitatively evaluated whether the proposed method is effective in estimating the self-position on the basis of the positions and orientations of the particles of MCL. Fig. 5 shows the positions and orientations of the particles. The left figure shows the positions and orientations of the particles of MCL that employed only odometry and distance sensor data. The right figure shows the positions and orientations of particles that reflected an object categorization result before making the particles convergent in a measure. The arrows in these figures denote the estimated positions and orientations of the particles. The experimental results show that the proposed method can reduce the number of particles that incorrectly estimate the self-position and corrects them to the vicinity of the actual position.

D. Estimated accuracy of self-position

In this study, we evaluated the proposed method by quantitatively comparing four methods. We conducted 14 trials in the experiment. We moved the robot to a specified point and made it estimate the self-location in each trial. Fig. 6 shows the points specified in the experiment and an obtained image. The arrows indicate the direction of the robot when it obtained an image in each trial. The colors of the arrows correspond to the colors of the arrows for each Gaussian distribution A, B, C, and D, shown in Fig. 4. Fig. 7 shows the floor plan of the room where we conducted the experiment. When the robot arrived at each specified point, it observed one image. Then, the robot reflected the result of Bag-of-Features representation of the feature vector that the proposed method or each comparative method extracted from the image. The position that a particle estimated is denoted as $\bar{\boldsymbol{x}}_t = (\bar{x}_t, \bar{y}_t, \bar{\sin}\theta_t, \bar{\cos}\theta_t)$, and the actual position is denoted as ${\boldsymbol{x}_t}^* = ({x_t}^*, {y_t}^*, \sin{\theta_t}^*, \cos{\theta_t}^*)$. The evaluation method was based on the estimation error of the particles as follows:

$$e_t = \Big\{(\bar{x}_t - x_t^*)^2 + (\bar{y}_t - y_t^*)^2 + (\bar{\sin}\theta_t - \sin\theta_t^*)^2 + (\bar{\cos}\theta_t - \cos\theta_t^*)^2\Big\}^{\frac{1}{2}}. \quad (13)$$

Fig. 8 shows the average estimation error of 14 trials. TABLE I lists the values of the estimation errors of 14 trials. Fig. 8 shows that the robot reduced the positional estimation error in the case where an object categorization result was used. Moreover, TABLE I shows that the proposed method estimated the self-location more accurately in many trials. This result enables us to confirm that the robot can reduce the global positional error by using an object categorization result obtained by using the CNN in many cases. However, the positional estimation error was relatively large in cases where the robot observed an image and used an object categorization result generated by the CNN in "region" $71-D$. We attribute this result to a similar object recognition result obtained by using the CNN. In the trial where the robot observed an image in "region" $71-D$, the image was categorized into the class of "sliding door" with a high probability by the CNN. Object categorization results with a high value of the class of "sliding door" are obtained by the CNN in some "regions," for instance, "region" $12-B$. Therefore, it is considered that particles estimate potions in "regions" such as "region" $12-B$ as self-localization. TABLE I shows that the comparative methods considered in this study can generate more positive results than the proposed method. These results suggest that the methods using SIFT and RGB color codes generate both good and

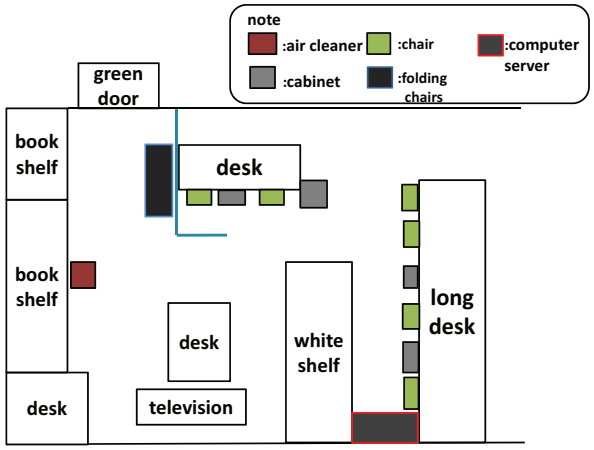

Fig. 7. Floor plan of the laboratory where the experiment was conducted

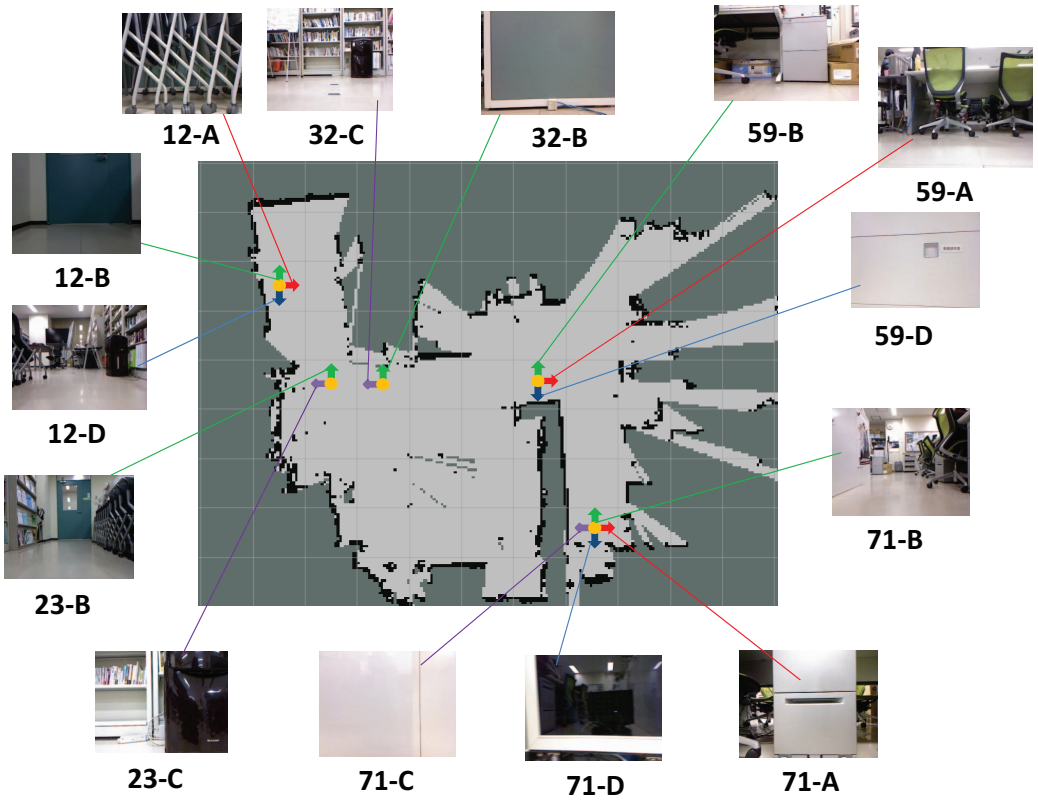

Fig. 6. A map is built by SLAM and the images observed in the experiment. Each yellow point denotes the position where the robot obtained an image and used it as an object recognition result for self-localization.

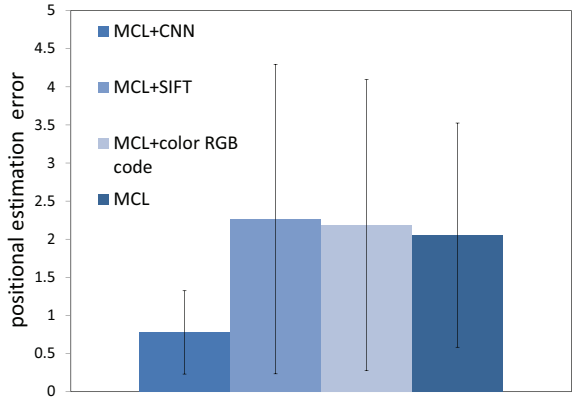

Fig. 8. A graph of the estimation error of the comparative methods.

TABLE I
RESULTS OF THE COMPARISON

	MCL +CNN	MCL +SIFT	MCL +RGB	MCL
12-A	**0.54**	**0.54**	6.85	6.02
12-B	**0.29**	3.87	0.96	3.91
12-D	0.99	0.99	**0.98**	1.44
23-B	**0.40**	**0.40**	0.63	0.47
23-C	0.40	**0.38**	0.60	1.9
32-B	**0.47**	0.56	0.89	0.80
32-C	0.52	**0.30**	2.31	2.04
59-A	**1.21**	5.28	2.69	1.33
59-B	**0.30**	**0.30**	1.02	0.38
59-D	1.28	1.33	**0.19**	1.21
71-A	**0.86**	5.81	2.73	2.12
71-B	**0.81**	3.49	2.05	1.34
71-C	**0.43**	3.49	5.82	3.47
71-D	2.39	4.97	2.89	**2.33**

bad results. For example, the method using RGB color codes reduces the estimation error grossly in the "region" where the robot observed images representing a simple color, such as $59-D$. Further, the method using SIFT generated a more positive result than the proposed method in the "region" where the robot could extract many feature points from the observed image, such as $32-C$. In contrast, the method using SIFT exhibited negative results for the "region" where the robot extracted few feature points from the observed images.

IV. CONCLUSION

In this paper, we proposed a self-localization method that exploited the object recognition results obtained by using

the CNN for autonomous vehicles. The experimental results showed that the proposed method was able to converge the positions and orientations of the particles and reduced the global positional errors. Moreover, the comparative experiment showed that a self-localization method using the object categorization results generated by the CNN locally, provided more positive results than the methods using SIFT and color RGB codes. In the future, we would like to let a robot estimate self-location by exploiting the results of multimodal integration and object categorization by using CNN, SIFT, and RGB color codes along with the Bag-of-Features representation.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

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

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[4] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1237.

[5] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8614–8618.

[6] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 253–256.

[7] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 639–655.

[8] N. Karlsson, E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. E. Munich, "The vslam algorithm for robust localization and mapping," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 24–29.

[9] S. Ahn, M. Choi, J. Choi, and W. K. Chung, "Data association using visual object recognition for ekf-slam in home environment," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2588–2594.

[10] G. Hu, S. Huang, L. Zhao, A. Alempijevic, and G. Dissanayake, "A robust rgb-d slam algorithm," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1714–1719.

[11] Y. Hagiwara, T. Inamura, and Y. Choi, "Effectiveness evaluation of view-based navigation for obstacle avoidance," in *Control, Automation and Systems (ICCAS), 2013 13th International Conference on*. IEEE, 2013, pp. 1029–1033.

[12] Y. Hagiwara, Y. Choi, and K. Watanabe, "An improved view-based navigation by adjustable position resolution," in *Autonomous Robots and Agents, 2009. ICARA 2009. 4th International Conference on*. IEEE, 2009, pp. 298–301.

[13] A.Taniguchi, H.Yoshizaki, T.Inamura, and T.Taniguchi, "Research on simultaneous estimation of self-location and location concepts," *System,Control and Information*, vol. 27, no. 4, pp. 166–177, 2014.

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.