2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Vision-based Navigation with Pose Recovery under Visual Occlusion and Kidnapping

Jungho Kim and In-So Kweon

Abstract— **Vision-based robotic applications such as Simultaneous Localization and Mapping (SLAM), global localization, and autonomous navigation have suffered from problems related to dynamic environments involving moving objects and kidnapping. One of the possible solutions to these problems is to establish robust correspondences when obtaining images from static scenes. Therefore we propose an efficient technique for determining correspondences to recover the current camera pose; in the proposed method, the FAST corner detector and SIFT descriptors are combined because in many methods for vision-based robotic applications, corner features have been adopted since they enable fast computation and simplify the computation of the correspondences between consecutive images. However, to recover the pose of the camera after kidnapping or at an unknown initial position, a robust feature matching algorithm is required because the pose of a camera is unlikely to be the same as the poses in the database images. For this purpose, first, we determine some candidates for correspondences by combining corners with their multiple descriptors computed from previously defined scales, and then we select one of these candidates by optimizing the scale using a variant of the mean-shift algorithm. We apply the proposed matching algorithm to kidnapping and visual occlusion problems in autonomous navigation.**

I. Introduction

Intelligent robots require the ability to perform desired tasks in the unstructured environment without continuous human guidance. For this purpose, robots necessitate the capability of autonomously arriving at the desired position. Most autonomous navigation techniques start from defining relations between the high-level environment perception and the low-level robot operation. For this task, vision sensors are especially efficient in that they provide more information on scene interpretation than range scanning sensors such as LRFs and sonar sensors. Vision-based autonomous navigation approaches can be classified into two categories based on the environment representation.

In the model-based approaches, the environment is represented by landmarks and descriptions of the image features which are involved in map building [1][2][3]. In the appearance-based approaches, these navigation methods have adopted the representation of the environment by composing of a set of images or variants of images [4][5][6][7][8].

However, for most vision-based navigation methods, images obtained from vision sensors are strongly dependent of environment conditions such as illumination conditions or visual occlusion by moving objects. Thus, when the images are influenced by moving objects or kidnapping, the previous navigation methods are prone to failure. To solve these problems, some previous studies have been proposed in the robotics community [9][10] and in the computer vision community [11][12][13] for SLAM and mapping problems.

To cope with these problems, the robust feature matching algorithm is required because a camera is unlikely to have the same pose with those of previously obtained images, that results in the large difference between two images. For this purpose, existing feature detection and description methods such as SIFT [14] and SURF [15] can be available. However, the high computational complexity of these methods prevents us from applying them to real-time robotic applications. Instead, in [11], the authors described a visual SLAM algorithm that is compatible with erratic camera motion and visual occlusion. In this approach, after the visual occlusion or camera shaking, they use the uncertainty of the camera pose and 3D scene points to predict scale changes in the images, and descriptors are built at multiple scales. However, this approach has an strong assumption that a camera is located near the place where it fails SLAM.

In this paper, we present an efficient method for recovering the camera pose under visual occlusion and kidnapping problems during vision-based autonomous navigation. For autonomous navigation, we also adopt our navigation system [3] based on the teaching and replay strategy [2][6] in which a robot is manually driven though the path once during the teaching step; subsequently the robot follows the path autonomously during the replay step.

This paper is organized as follows. In Section II, our navigation system is briefly presented. In Section III, the pose recovery algorithm is decribed to solve kidnapping and visual occlusion problems. Section IV shows various experimental results on scene recognition and vision-based navigation under kidnapping and moving objects to demonstrate feasibility of the proposed method.

II. Autonomous Navigation

A. Teaching Step

Before autonomous navigation, we construct the global map and store representative images by using visual odometry that consists of a few sub-parts, as shown in Fig. 1.

For each stereo pair, we first extract FAST corner points [16] in the left image and then apply the modified KLT feature tracker [3] to stereo images to obtain correspondences. The 3D coordinates of the matched corner points are

Jungho Kim is with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, ,Korea jhkim@rcv.kaist.ac.kr

In-So Kweon is a Professor of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Korea iskweon@kaist.ac.kr

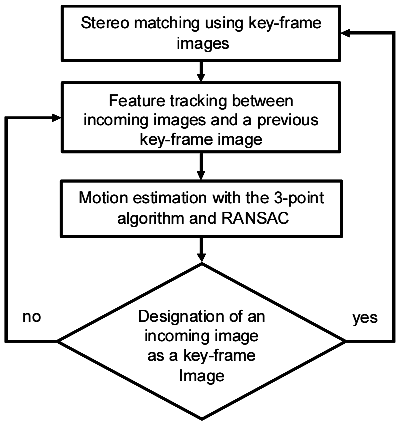

Stereo matching using key-frame images
Feature tracking between incoming images and a previous key-frame image
Motion estimation with the 3-point algorithm and RANSAC
Designation of an incoming image as a key-frame Image
no
yes

Fig. 1. Overall procedure for the teaching step

used for map building and for motion estimation. For motion estimation we use 'visual odometry' [17] that estimates the movement of a stereo head in real time.

The number of tracked corners between an incoming image and a previous key-frame image is a measure to determine the key-frame places. If the number of points tracked by the KLT tracker [18] is smaller than a pre-defined threshold, we designate an incoming image as a key-frame image and estimate the relative pose w.r.t the previous key-frame. During autonomous navigation the key-frame images and the feature-based map computed from motion estimation and stereo matching are used.

B. Replay Step

For home to destination locations, we use the coordinates of the epipole [19]. The epipole is the image in one view of the camera center of the other view. Thus, if a camera faces a destination location, the epipole must be located near a principal point in the image. Because we know camera poses of two views, we compute the epipole using the following way, as given in Eq. (1) [19].

$$\mathbf{e} = \mathbf{K}[\mathbf{R}_c \quad \mathbf{t}_c]\mathbf{X}_d \tag{1}$$

where $\mathbf{X}_d$ is the destination location where a key-frame image was captured, and $\mathbf{R}_c$ and $\mathbf{t}_c$ represent the current pose of the left camera. $\mathbf{K}$ represents the camera matrix.

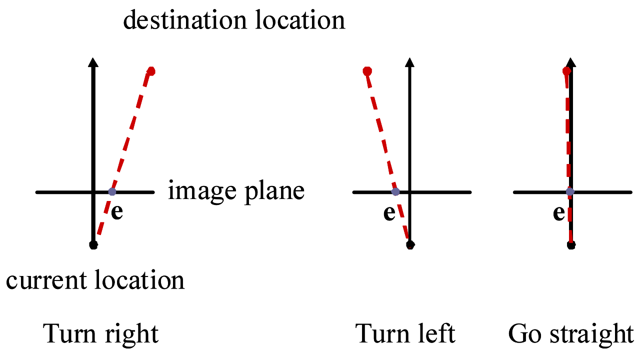


Fig. 2. Navigation strategy based on epipole coordinate

If the x-coordinate of the epipole, e_x, is located at the right side of the principal point, fc_x, then a robot turns right, as shown in Fig. 2. We can summarize the navigation strategy, as stated in Eq. (2) and Eq. (4). If a camera is not close to the destination location, a robot moves according to Eq. (2).

$$\begin{array}{lcl} \text{if} \;\; e_x > fc_x + t_x & : & \text{turn right} \\ \text{else if} \;\; e_x < fc_x - t_x & : & \text{turn left} \\ \quad \text{else} & : & \text{go straight} \end{array} \tag{2}$$

where t_x is a positive value.

To determine whether a robot reaches a destination place, we compare locations of the robot by Eq. (3).

$$d = |R_d^T t_d - R_c^T t_c| \tag{3}$$

where $-R_d^T t_d$ stands for the location of destination with respect to the global coordinate and $-R_c^T t_c$ means the location of the current camera computed by visual odometry.

When a robot reaches the destination location, we adjust the robot orientation by correspondences between the current image and one of key-frame images to satisfy the relation given in Eq. (4). To determine the correspondences, we use the feature matching method [3] that utilizes the 3D coordinates of the landmarks in the map and the current camera pose.

$$\begin{array}{lcl} \text{if} \;\; \hat{x}_c > \hat{x}_d + t_a & : & \text{turn left} \\ \text{else if} \;\; \hat{x}_c + t_a < \hat{x}_d & : & \text{turn right} \\ \quad \text{else} & : & \text{stop} \end{array} \tag{4}$$

where t_a is a marginal angle and $\hat{x}_d$ and $\hat{x}_c$ are means of x coordinates among correspondences between key and current images, respectively.

Before a robot stops according to Eq. (4), which means a robot has the same pose with those of key-frame images, we use visual odometry using a stereo camera to localize a robot by estimating the frame-to-frame relative poses and integrating them over time. Despite of environment changes, *e.g.*, some objects newly appeared or some were removed from the environment, a robot can navigate through those environments because it estimates its pose using visual odometry instead of landmark-based localization.

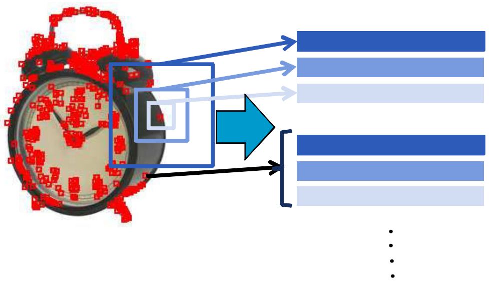
When a robot stops according to the conditions stated in Eq. (4), we regard this operation as home to the destination location and designate the next key-frame image as the destination image.

Since the incremental-motion-based visual odometry alone will drift eventually as the error accumulates, it is only locally accurate. To cope with this problem, the navigation system returns to landmark-based localization when it satisfies the condition of 'stop' stated in Eq. (4). We recompute the current camera pose by using the 3-point algorithm [20] followed by LM optimization [21] after finding correspondences between 3D landmarks in the global map and their image coordinates.

III. Global Localization Step

Due to severe visual occlusion and kidnapping, the number of tracked features drastically decreases and a robot cannot continue to navigate. In these cases, a robot recovers its pose using global localization when obtaining the images from the static scenes. A robot moves backward until it succeeds in recovering its pose.

(a) original image



(b) FAST corner extraction and description by using previously defined 3 scales (s_1, s_2, and s_3)

Fig. 3. Feature Detection and Descriptrion

A. Scene Recognition

In scene recognition, the image in the database that corresponds to a query image is determined. Because the pose of a robot is unlikely to be identical to the poses in the database images, scene recognition must be robust to view variations and illumination changes. Moreover, to localize a robot, we have to establish correspondences between 3D points computed from the corners and their coordinates in the incoming images. In our approach, we extract corners using the FAST detector [16]; this detector outperforms other detectors in terms of the computational cost and repeatability. We also compute three SIFT descriptors for each corner by using previously defined scales (16 $\times$ 16, 32 $\times$ 32 and 48 $\times$ 48 patches centered at the corner), as shown in Fig. 3, because the FAST detector does not provide scales for computing descriptors. SIFT descriptors have shown to give reliable feature matching over a wide range of view and illumination conditions. We can reduce the computational cost for computing the SIFT descriptors by constructing the integral images. The integral images have been used for face detection and feature detection to reduce the computation time, as introduced in [22][15]. First, we construct 8 integral images, and each integral image is computed by

$$J_i(x,y) = \sum_{u \le x} \sum_{v \le y} \delta(\phi(u,v) - \theta_i) \tag{5}$$

where $\delta(x)$ represents the delta function which returns 1 if x is zero, otherwise returns 0. θ_i is computed by dividing the 360 degree range of orientations into 8 bins, and $\phi(u,v)$ is a quantized orientation that is computed at a pixel (u,v) based on 8 bins.

We compute a SIFT descriptor from 16 sub regions that are equally divided by a region defined by the center location (x,y) and the scale s, as shown in Fig. 4, by

$$\begin{aligned} h(8m+k-8) = {} & w_m J_k(x_r(m), y_b(m)) - w_m J_k(x_l(m), y_b(m)) \\ & - w_m J_k(x_r(m), y_t(m)) + w_m J_k(x_l(m), y_t(m)) \end{aligned} \tag{6}$$

where $m = 1,2,\cdots,16, k = 1,2,\cdots,8$, w_m is a weight for a sub region r_m. $(x_l(m), y_t(m))$ and $(x_r(m), y_b(m))$ represent top-left and bottom-right coordinates of a sub region r_m, respectively. Each bin is normalized by $h(i) = h(i)/\sum_j h(j)$

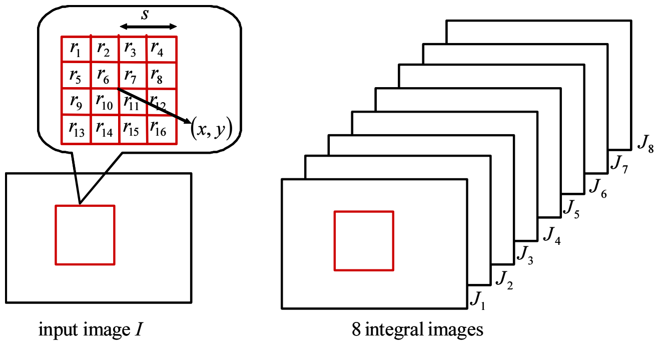

s
r_1 r_2 r_3 r_4
r_5 r_6 r_7 r_8
r_9 r_{10} r_{11} r_{12}
r_{13} r_{14} r_{15} r_{16}
(x,y)
J_1 J_2 J_3 J_4 J_5 J_6 J_7 J_8
input image I
8 integral images

Fig. 4. Computation of SIFT descriptors by using integral images

Even though we compute the multiple SIFT descriptors (two more descriptors) for each corner, the computation does not require much time because in this algorithm, a majority of the computation time is devoted to computing integral images [1]. We assume that robot navigation does not encounter the cases in which images are deformed by camera rotation about the optical axis because we cannot compute SIFT descriptors for arbitrary image rotations when using the integral images. For efficient feature matching with many descriptors in the database, we adopt an image-search method proposed by Nister et al. [23]. In this method, all SIFT descriptors are arranged in a vocabulary tree using the K-means clustering algorithm [24]. For matching, each descriptor is simply propagated down the tree at each level by comparing it with the k clustering center vectors. Before the replay step, we construct the vocabulary tree from the SIFT descriptors that correspond to 3D landmarks computed from FAST corners in all key-frame images. For scene recognition, we also extract FAST corners and three SIFT descriptors for each corner from a query image with same scales. During the computation of three descriptors for each corner, we inspect

[1]In our implementation, the computational costs for computing 150 and 1800 descriptors using the integral images are 0.078 sec and 0.080 sec, respectively. (The image resolution is 640 $\times$ 480 pixels.)

the sensitivity to scale variation by computing σ_h that is used for scale optimization in the next step and selection of multiple candidates, as the following mean distance:

$$\sigma_h = \frac{d(s_1, s_2) + d(s_2, s_3)}{2} \tag{7}$$

where $d(s_i, s_j) = \left\| h_t(s_i) - h_t(s_j) \right\|$

Using the vocabulary tree, we select a maximum of ten descriptors[2] that may correspond to each feature of the query image by the following matching threshold:

$$d_{th} = d_0 + \sigma_h \tag{8}$$

where d_0 is the matching threshold for SIFT descriptors.

B. Scale Optimization

In the scene recognition stage, we select a maximum of ten candidates among all the descriptors in a vocabulary tree based on roughly defined scales. In this stage, we compute the optimal scale with given a scale of the feature in the database on the basis of the mean-shift algorithm [25][26].

We define the problem of estimating the optimal scale and the corresponding corner in the database that maximizes the following posterior distribution as follows:

$$\underset{s}{\operatorname{argmax}}\, p(s|s_d) \tag{9}$$

For each candidate s_d, we represent the distribution of the scale by a mixture of Gaussians as follows:

$$f(s) = p(s|s_d) = \alpha \sum_i w_i \exp\left(-\frac{(s - s_i)^2}{2\sigma_s^2}\right) \tag{10}$$

where $\alpha = \frac{1}{\sum_i w_i}$ and σ_s is a parameter depending on the defined scales.

w_i is a weight for the ith Gaussian distribution; w_i is obtained by comparing the current descriptor corresponding to s_i with a descriptor for a match candidate s_d, as shown in Eq. (11)

$$w_i = \exp\left(-\frac{\|h_t(s_i) - h_d(s_d)\|^2}{2\sigma_h^2}\right) \tag{11}$$

where $h_d(s_d)$ is the SIFT descriptor which is selected as a candidate.

This kind of a mixture of Gaussians can be optimized by employing the mean-shift algorithm, a simple iterative procedure in which each data point is shifted to the average of data points.

By taking the estimate of the density gradient as the gradient of the density estimate, we have

[2]When constructing the vocabulary tree, we assigned ten nearest descriptors to each descriptor.

$$\nabla f(s) = \frac{\alpha}{\sigma_s^2} \sum_i (s_i - s)\, w_i \exp\left(-\frac{(s - s_i)^2}{2\sigma_s^2}\right) =$$
$$\frac{\alpha}{\sigma_s^2}\left[\sum_i w_i \exp\left(-\frac{(s - s_i)^2}{2\sigma_s^2}\right)\right]\left[\frac{\sum_i s_i w_i \exp\left(-\frac{(s-s_i)^2}{2\sigma_s^2}\right)}{\sum_i w_i \exp\left(-\frac{(s-s_i)^2}{2\sigma_s^2}\right)} - s\right] \tag{12}$$

where $\sum_i w_i \exp\left(-\frac{(s-s_i)^2}{2\sigma_s^2}\right)$ can be assumed to be nonzero. Therefore, the final pair of brackets in Eq. (12) contains the sample mean-shift vector, $m_v(s)$.

$$m_v(s) = \frac{\sum_i s_i w_i \exp\left(-\frac{(s-s_i)^2}{2\sigma_s^2}\right)}{\sum_i w_i \exp\left(-\frac{(s-s_i)^2}{2\sigma_s^2}\right)} \tag{13}$$

The mean shift procedure is defined recursively by computing the mean shift vector $m_v(s)$ and translating the center of Gaussian kernel by $m_v(s)$.

Let us denote by $\{y_j\}_{j=1,2,\ldots}$ the sequence of successive scales that are iteratively estimated, where

$$y_{j+1} = \frac{\sum_i w_i s_i \exp\left(-\frac{(y_j-s_i)^2}{2\sigma_s^2}\right)}{\sum_i w_i \exp\left(-\frac{(y_j-s_i)^2}{2\sigma_s^2}\right)} \tag{14}$$

Initially, we have 3 samples (scales) that are not sufficient to compute the optimal scale. Thus we update the distribution, defined in Eq. (10) as follows:

$$f(s) = \beta\left[\sum_i w_i \exp\left(-\frac{(s - s_i)^2}{2\sigma_s^2}\right) + \sum_j v_j \exp\left(-\frac{(s - y_j)^2}{2\sigma_s^2}\right)\right] \tag{15}$$

where β is a normalizing term. $w_i = \exp\left(-\frac{(h_t(s_i)-h_d(s_d))^2}{2\sigma_h^2}\right)$ and $v_j = \exp\left(-\frac{(h_t(y_j)-h_d(s_d))^2}{2\sigma_h^2}\right)$.

Based on Eq. (15) that uses newly computed scales and their descriptors during the optimization step, we compute the next optimal scale as follows:

$$y_{t+1} = \mathbf{round}\left(\frac{\sum_{i=1}^{3} s_i w_i \exp\left(-\frac{(y_t-s_i)^2}{2\sigma_s^2}\right) + \sum_{j=1}^{t} y_j v_j \exp\left(-\frac{(y_t-y_j)^2}{2\sigma_s^2}\right)}{\sum_{i=1}^{3} w_i \exp\left(-\frac{(y_t-s_i)^2}{2\sigma_s^2}\right) + \sum_{j=1}^{t} v_j \exp\left(-\frac{(y_t-y_j)^2}{2\sigma_s^2}\right)}\right) \tag{16}$$

where '**round**' means rounding off to the nearest integer because we compute descriptors using integral images. We initially set y_0 to the scale that corresponds to the maximum weight among three scales.

The mean shift iteration is terminated when one of the below criterions is satisfied.

- there is no variation on the scale. i.e. $y_{t+1} = y_t$
- the newly computed descriptor has larger error than previous one. i.e. $v_{t+1} < v_t$, we replace y_{t+1} with y_t.
- it reaches at the maximum number of iterations.

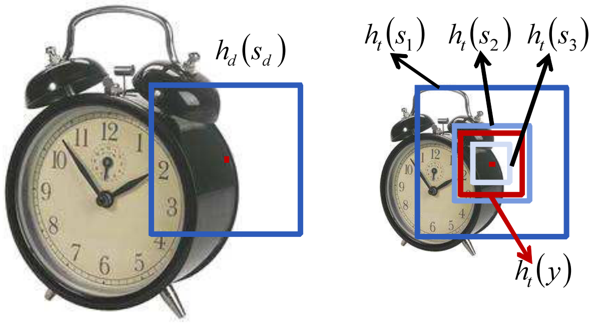


Fig. 5. Scale optimization using given 3 descriptors ($h_t(s_1), h_t(s_2)$, and $h_t(s_3)$)

After terminating mean shift procedure for each corner, we determine whether two points match or not by

$$\|h_t(y_{t+1}) - h_d(s_d)\| < d_0 \tag{17}$$

C. *Designation of New Destination*

A robot autonomously recovers the current pose by using the 3-point algorithm and LM optimization with the correspondences. If there exist enough inliers after estimating the current pose, then a robot designates the next destination according to its current pose as shown in Eq. (18). For example, if a robot computes its pose, as shown in Fig. 6 then two angles α and β are measures of deciding the next destination node. α represents how much a robot should rotate and β measures overlapped image regions after reaching at the destination.

$$n = \underset{k \in [i-\sigma, i+\sigma]}{\text{argmax}} \ (cos\alpha_k cos\beta_k) \tag{18}$$
$$\text{only if} \ \ cos\alpha_k > 0, cos\beta_k > 0$$

Here, n is the index of the next destination, i is the index of the key-frame image corresponding to the maximum posterior, and σ is a search range.

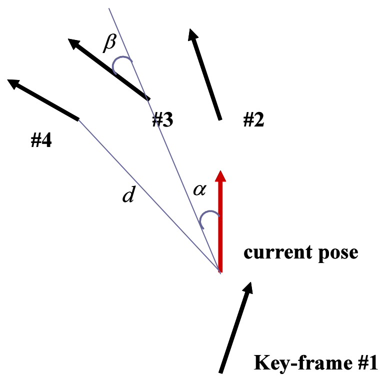


Fig. 6. Decision rule of the next destination from the recovered camera pose

TABLE I
MEAN OF PROCESSING TIME FOR SCENE RECOGNITION WITH 19397 CORNER FEATURES IN THE DATABASE (2.4GHZ CPU, 320 × 240 PIXELS)

	processing time for scene recognition (ms)
SIFT	205
Proposed	46

TABLE II
COMPUTATIONAL COST FOR NAVIGATION WITHOUT GLOBAL LOCALIZATION (2.4GHZ CPU, 320 × 240 STEREO IMAGES)

total time (ms)	the number of frames	average time (ms)
69672	2201	31.655

IV. EXPERIMENTAL RESULTS

A. *Scene Recognition*

Table I lists the mean computational costs for computing correspondences between corners in an incoming image and 19397 corners computed from 43 database images during the evaluation of 200 query images.

Fig. 7 shows one example of feature matching performed using 43 database images.

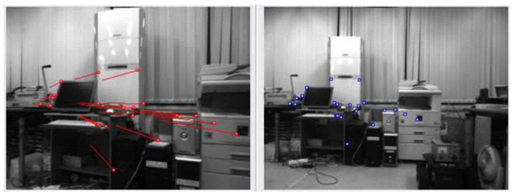
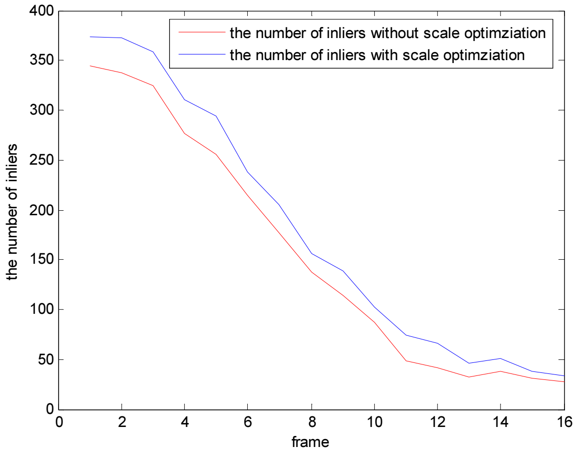


Fig. 7. Feature matching result with 43 database images

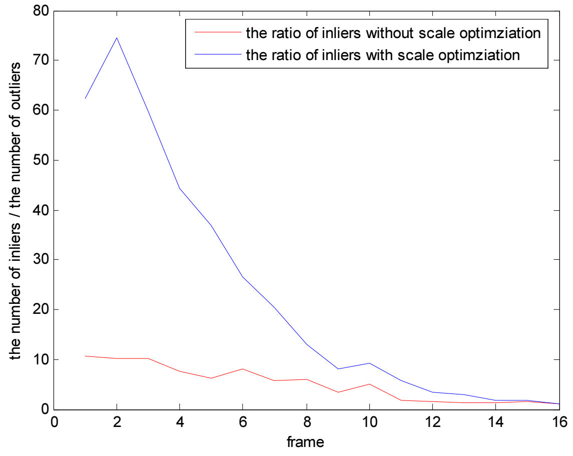
For performance evaluation, we select 10 database images that are left images in stereo and 16 test images among the right images that correspond to the first database image. Fig. 8(a) shows the number of inliers over a sequence of test images. The red line is obtained by feature matching without scale optimization, and the blue line shows the number of inliers when using scale optimization. Fig. 8(b) shows the ratio between the number of inliers and the number of outliers. By using scale optimization, we can obtain more inliers and effectively suppress outliers.

B. *Navigation in Dynamic Environments*

The proposed navigation algorithm was implemented on a laptop that controlled a mobile robot mounted with a stereo camera. Our system runs in real time and it processes 320×240 images captured from a stereo camera. Table II shows the computational cost for navigation involving map building and motion estimation after capturing 2201 frames using the same processor without global localization. To evaluate the efficiency of the proposed method, a robot was manually driven in an environment during the teaching step when there were no people as shown in Fig. 9(b). After a few hours, when some people arrived at the office environment and they wandered, the robot autonomously navigated through the environment. During navigation, a robot was moved compulsively to another location as shown in Fig. 9(c). Fig. 9(a) shows both key-frame locations (red



(a) The number of inliers



(b) The ratio between inliers and outliers (the number of inliers / the number of outliers)

Fig. 8. Performance comparison for scene recognition without scale optimization and with scale optimization

dots) overlapped with the generated map. Even when there exist walking people, sudden motion and kidnapping, a robot can follows a desired using the proposed method.

V. Conclusion

We have presented a vision-based navigation system that is robust to kidnapping and moving objects. For this purpose, we proposed an efficient pose recovery method that combines FAST corners and their multiple SIFT descriptors; first, we determine some candidates for correspondences by combining corners with their multiple descriptors computed from previously defined scales, and then we select one of these candidates by optimizing the scale using a variant of the mean-shift algorithm. The proposed matching algorithm can be used for many vision-based robotic applications. We demonstrate the robustness of our navigation system to the various conditions.

VI. ACKNOWLEDGMENTS

This work was supported by National Strategic R&D Program for Industrial Technology, Korea. We would like to express our sincere gratitude to the reviewers of this work for their valuable comments and recommendations.

References

[1] S. Se, D. Lowe and J. Little, Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmakrs, *IJRR*, vol. 21, 2002.

[2] S. Ŝegvić, A. Remazeilles, A. Diosi and F. Chaumette, Large scale vision-based navigation without an accurate global reconstruction, *CVPR*, 2007.

[3] J. Kim, Y. Bok and I. S. Kewon, Robust Vision-based Navigation against Environment Changes, *IROS*, 2008.

[4] Y. Yagi, K. Imai, K. Tsuji and M. Yachida, Iconic Memory-Based Omnidirectional Route Panorama Navigation, *PAMI*, vol. 27, 2006.

[5] J. Gaspar, N. Winters and J. Santos-Victor, Vision-based Navigation and Environmental Representations With an Omni-directional Caemra, IEEE Transactions on Robotics and Automation, vol. 16, 2000.

[6] Z. Chen and S. T.Birchfield, Qualitative Vision-based Mobile Robot Navigation, *ICRA*, 2006.

[7] O. Booij, B. Terwijn, Z. Zovkovic and B. Kröse, Navigation using an appearance based topological map, *ICRA*, 2007.

[8] F. Faundorfer, C. Engles and D. Nistér, Topological mapping, localization and navigation using image collections, *IROS*, 2007.

[9] D. F.Wolf and G. S. Sukhatme, Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments, Autonomous Robots, vol. 19, no. 1, 2005.

[10] C.-C. Wang, C. Thorpe and S. Thrun, Online Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects, *ICRA*, 2003.

[11] D. Chekhlov, M. Pupilli, W. Mayol and A. Calway, Robust Real-Time Visual SLAM Using Scale Prediction and Exemplar Based Feature Description, *CVPR* , 2007.

[12] B. Williams, G. Klein and I. Reid, Real-Time SLAM Relocalisation, *ICCV*, 2007.

[13] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit and P. Fua, View-based maps, *RSS*, 2009.

[14] D. Lowe, Distinctive image features from scaleinvariant Keypoints, *IJCV*, vol. 60, no. 2, 2004.

[15] H. Bay, T. Tuytelaars and L. V. Gool, SURF:Speeded Up Robust Features, *ECCV*, 2006.

[16] E. Rosten, R. Porter and T. Drummond, Machine Learning for High-Speed Corner Detection, *ECCV*, 2006.

[17] D. Nistér and O. Naroditsky and J. Bergen, Visual Odometry, *CVPR*, 2004.

[18] J. Shi and C. Tomasi, Good Features to Track, *CVPR*, 1994.

[19] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, ISBN 0-521-62304-9, 2000.

[20] R. Haralick , C. Lee, K. Ottenberg and M. Nölle, Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem, *IJCV*, vol. 13, no. 3, 1994.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in C, pp. 683–688, Second Edition, Cambridge Univerisity Press, 2002.

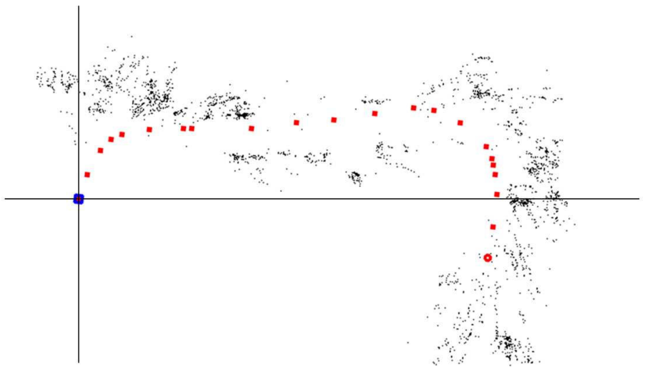
[22] P. Viola and M. Jones, Rapid Object Detection using a Boosted Cascade of Sample Features, *CVPR*, 2001.

[23] D. Nistér and H. Stewenius, Scalable Recognition with a Vocabulary Tree, *CVPR*, 2006.

[24] R. O. Duda and P. E. Hart and D. G. Stork, Pattern Classifcation, Wiley-Interscience Press.

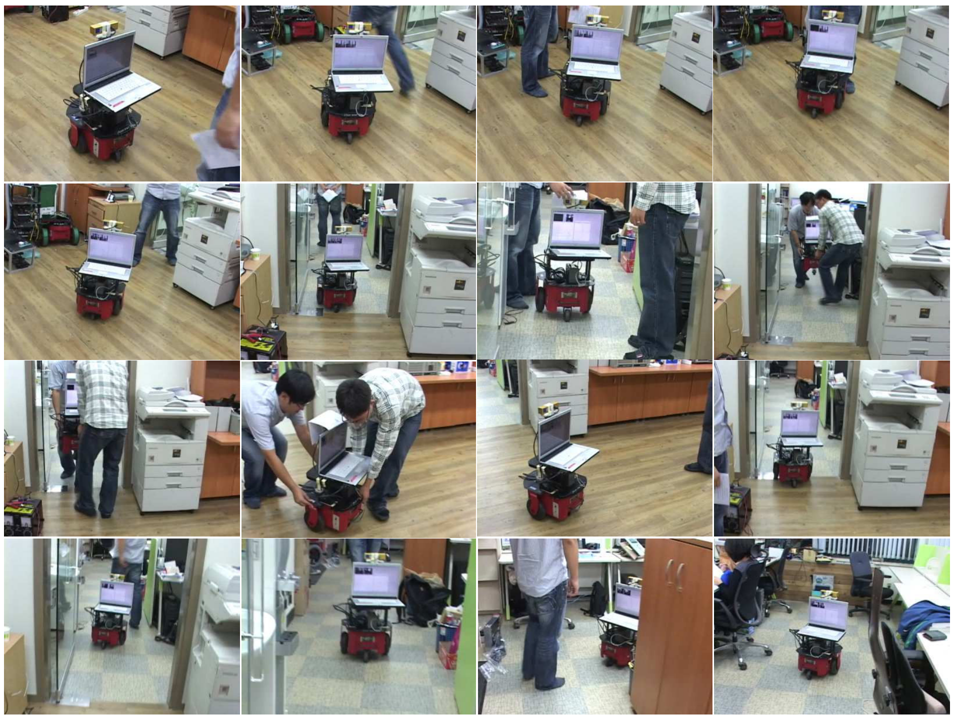
[25] K. Fukunaga, Introduction to Statistical Pattern Recognition, Second Ed., Academic Press, Boston, 1990.

[26] Y. Cheng, Mean Shift, Mode Seeking, and Clustering, *PAMI*, vol. 17, 790–799 (1995).



(a) The global map and key-frame locations obtained during the teaching step

(b) Some key-frame images taken at the office when there were no people



(c) Representative snapshots from a video of navigation at the office

Fig. 9. Navigation result for dynamic environments