Depth-hybrid speeded-up robust features (DH-SURF) for real-time RGB-D SLAM

Donghwa Lee[1a], Hyungjin Kim[2b], Sungwook Jung[2c] and Hyun Myung[*2]

[1]*Division of Computer & Communication Engineering, Daegu University, Gyeongsan, Republic of Korea*
[2]*Urban Robotics Laboratory, Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea*

(Received November 10, 2017, Revised November 21, 2017, Accepted December 22, 2017)

Abstract. This paper presents a novel feature detection algorithm called depth-hybrid speeded-up robust features (DH-SURF) augmented by depth information in the speeded-up robust features (SURF) algorithm. In the keypoint detection part of classical SURF, the standard deviation of the Gaussian kernel is varied for its scale-invariance property, resulting in increased computational complexity. We propose a keypoint detection method with less variation of the standard deviation by using depth data from a red-green-blue depth (RGB-D) sensor. Our approach maintains a scale-invariance property while reducing computation time. An RGB-D simultaneous localization and mapping (SLAM) system uses a feature extraction method and depth data concurrently; thus, the system is well-suited for showing the performance of the DH-SURF method. DH-SURF was implemented on a central processing unit (CPU) and a graphics processing unit (GPU), respectively, and was validated through the real-time RGB-D SLAM.

Keywords: speeded-up robust feature (SURF); depth-hybrid; red-green-blue depth (RGB-D) sensor; simultaneous localization and mapping (SLAM)

1. Introduction

The problem of finding correspondences in images from different views is an important issue in the computer vision field, including structure from motion (SfM) and visual simultaneous localization and mapping (SLAM) (Henry *et al.* 2012, Endres *et al.* 2012, Scaramuzza and Fraundorfer 2011, Szeliski 2010, Lee *et al.* 2012a, b, Kim *et al.* 2013, 2014, Park and Park 2014, Havangi *et al.* 2014, Chen and Jia 2014, Lee and Myung 2014, Lee and Oh 2015). Recently, keypoint feature detection and matching algorithms such as scale-invariant feature transform (SIFT) and speeded-up robust features (SURF) have become popular because of their invariance under scale and rotation transformations. These algorithms mainly consist of two parts: *detection*

∗Corresponding author, Associate Professor, E-mail: hmyung@kaist.ac.kr
[a]Assistant Professor, E-mail: leedonghwa@daegu.ac.kr
[b]Ph.D. Student, E-mail: hjkim86@kaist.ac.kr
[c]Ph.D. Student, E-mail: sungwook87@kaist.ac.kr

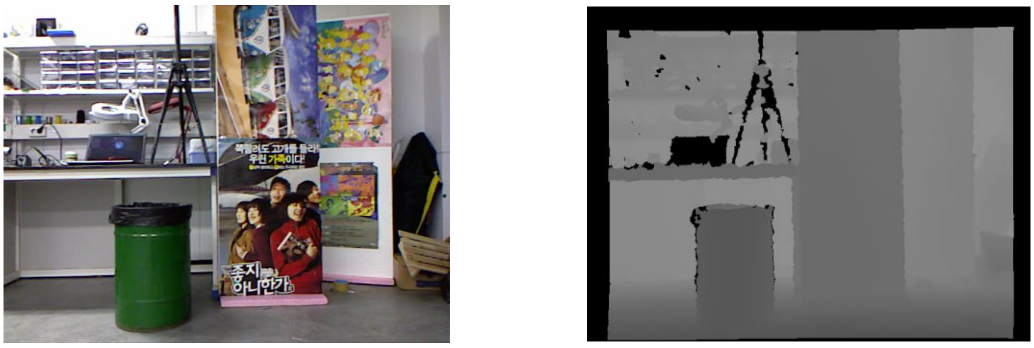

(a) (b)

Fig. 1 RGB-D sensor data, (a) RGB color image and (b) Per-pixel depth data

and *description* of keypoints. In the detection part, the variation of the standard deviation in the Gaussian kernel is used for scale-invariant feature points. By multiple doubling of the standard deviation σ, scale groups can be used to find keypoints in a variety of scale spaces. The scale group is referred to as an *octave*. In the SIFT algorithm, each octave is obtained by taking every second pixel from the image used in the previous octave, which is equal to doubling σ. To accelerate computation, SURF uses box filters based on a Hessian matrix to produce effects similar to the Gaussian kernel. In the SURF algorithm, changing the size of the box filters has the same effect as applying a different σ value. However, the presence of multiple octaves is a major cause of increased computational burden for feature extraction (Bay *et al.* 2008, Lowe 2004).

The last few years have seen the rapid development of 3D mapping technologies with the advent of inexpensive sensors such as the Microsoft Kinect. The Kinect sensor contains a depth sensor and color camera. The depth sensor provides depth data using the infrared (IR) projection method (Zhang 2012). Figs. 1(a) and 1(b) show a color image and the corresponding depth data from a Kinect sensor. Kinect-style sensors are called red-green-blue depth (RGB-D) cameras since they give color images and depth data concurrently. Recently, using this sensor, RGB-D SLAM systems have been developed. The RGB-D SLAM systems use a feature extraction method and depth data concurrently and give a corrected robot trajectory and a 3D point cloud map through loop closure and optimization techniques (Henry *et al.* 2012, Endres *et al.* 2012). Since the feature extraction and matching consume most of the processing time, improving performance of the method increases efficiency of the RGB-D SLAM systems.

We present a novel approach for fast keypoint detection using the RGB-D sensor. In the conventional methods, multiple octaves are necessary in order to counteract the keypoint scale. However, our method simply applies a different octave to each image pixel according to the octave masks generated from depth data. Therefore, the depth data helps to reduce the number of octaves while maintaining the scale-invariance property. The proposed method was implemented on a central processing unit (CPU) and a graphics processing unit (GPU), respectively, and was validated through the real-time RGB-D SLAM.

2. Proposed method

The main equations for keypoint detection with a Gaussian kernel are as follows

$$L(\mathrm{x},\sigma) = G(\mathrm{x},\sigma) * I(\mathrm{x}) \tag{1}$$

$$DoG(\mathrm{x}, k^m\sigma_s) = L(\mathrm{x}, k^{m+1}\sigma_s) - L(\mathrm{x}, k^m\sigma_s) \tag{2}$$

where $\mathrm{x} = (x, y)$; x and y are indices of column and row, respectively, I is an input image, G is a Gaussian function with a standard deviation σ, $*$ denotes a convolution operator, *DoG* (difference of Gaussian) is obtained by the difference in L s having σ_s multiplied by k^m and k^{m+1},respectively, where s denotes an octave scale, σ_s denotes a standard deviation in the s-th octave, k is a constant factor, and m is its index in the s-th octave. Keypoints are calculated from the local extrema of the *DoG*s with the variation of the standard deviation. One octave uses one σ for its keypoint. In order to find different-scaled keypoints, σ of the next octave is set as double of the previous on, i.e., $\sigma_s = 2\sigma_{s-1}$. The multi-octave strategy is useful for finding keypoints of various sizes on a 2D image as shown in Fig. 2. In the classical SURF algorithm, this scheme is applied using the Hessian matrix

$$\mathbf{H}(\mathrm{x},\sigma) = \begin{bmatrix} L_{xx}(\mathrm{x},\sigma) & L_{xy}(\mathrm{x},\sigma) \\ L_{xy}(\mathrm{x},\sigma) & L_{yy}(\mathrm{x},\sigma) \end{bmatrix} \tag{3}$$

where L_{xx}, L_{xy}, and L_{yy} are second-order derivatives, and they are approximated as box filters. Changing the size of the box filters represents the variation of the standard deviation (for details, see Bay *et al.* (2008), Lowe (2004)).

Generally, larger scale feature points are extracted from objects at closer distance of depth. When a high octave level is used, a large-scale keypoint is obtained. Bearing this in mind, by incorporating depth data, the keypoints can be obtained according to the distance to the keypoints. Using this scheme, we propose a keypoint detection method that uses an octave mask depending on depth data as shown in Fig. 3(b) unlike classical SURF algorithm as shown in Fig. 3(a). The proposed method is represented as follows

$$DoG_M\big(\mathrm{x}, k^m\sigma_s, M^s(\mathrm{x})\big) = \begin{cases} L(\mathrm{x}, k^{m+1}\sigma_s) - L(\mathrm{x}, k^m\sigma_s), & \text{if } M^s(\mathrm{x}) = 1 \\ 0, & \text{if } M^s(\mathrm{x}) = 0 \end{cases} \tag{4}$$

where $M^s(\mathrm{x})$ denotes the s-th octave mask, which can be 0 or 1. If $M^s(\mathrm{x}) = 0$, *DoG* process is skipped to improve computation time. For generation of $M^s(\mathrm{x})$, depth data should be handled at first. Because of the sensor system, the original depth data contains unmeasured data as shown in black spots in Fig. 4(a). Therefore, an image inpainting technique (Telea 2004) is employed to fill the unmeasured depth data as shown in Fig. 4(b). The image inpainting technique is widely utilized to restore small damaged portions of an image. The algorithm fills in blanks of depth data using a normalized weighting function with known neighborhood data. The algorithm is very simple to implement and runs fast using the fast marching method (FMM) described in Sethian (1996). After inpainting unmeasured depth data, the octave masks are generated according to distance information as follows

$$M^s(\mathrm{x}) = \begin{cases} 1, \text{if } d_s - \sigma_{d_s} \le D(\mathrm{x}) \le d_{s-1} + \sigma_{d_{s-1}} \\ 0, \quad \text{otherwise} \end{cases} \tag{5}$$

where $D(\mathrm{x})$ returns per-pixel depth data in x, $d_s = 2 \cdot d_{s-1}$, where d_s is a conditional distance value to determine the octave levels, and σ_{d_s} denotes the standard deviation of depth value at d_s according to the sensor system characteristics. While a lower octave mask is set to 1 in far distances, a higher octave mask is set to 1 in close distances. To reduce ambiguity depending on its sensor noise at the boundary point, d_s, of the octave mask, a marginal gap is generated by a standard deviation σ_{d_s} to overlap the octave masks at the boundary point. Since the depth data are fused into the SURF algorithm, we call this method as depth-hybrid SURF (DH-SURF). Fig. 5 shows DH-SURF extracting process using the generated masks from the depth value. The features are extracted from each octave according to the different octave masks. By combining features from different octaves, DH-SURF is performed. The depth data help reduces the number of octaves and redundant keypoints while maintaining the scale-invariance property. Furthermore, a decrease of keypoints leads to reduced processing time.

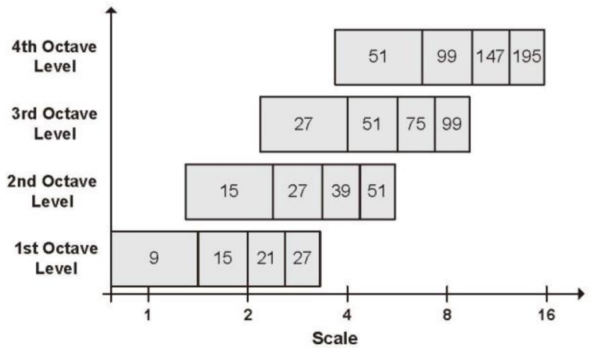
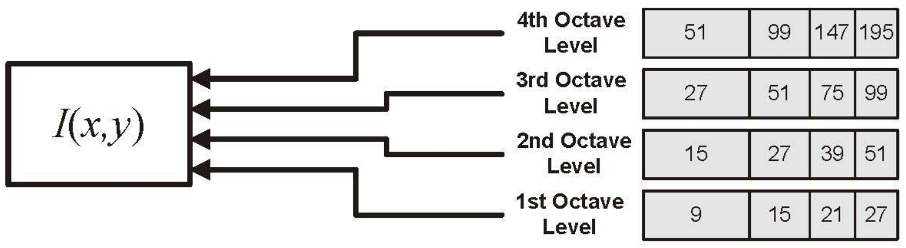
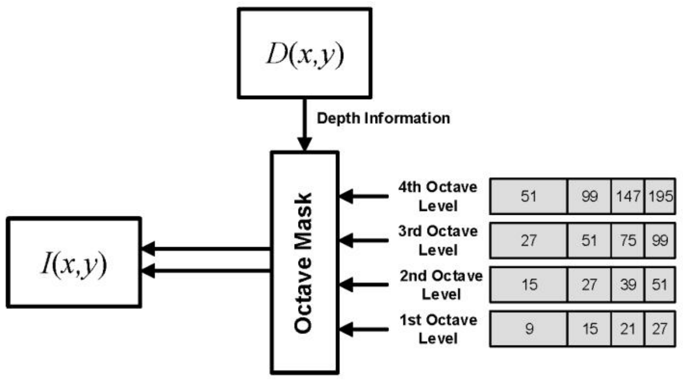


Fig. 2 Box filter sizes for different octave levels in SURF algorithm



(a) Classical SURF algorithm



(b) DH-SURF algorithm

Fig. 3 Overview of applying box filters

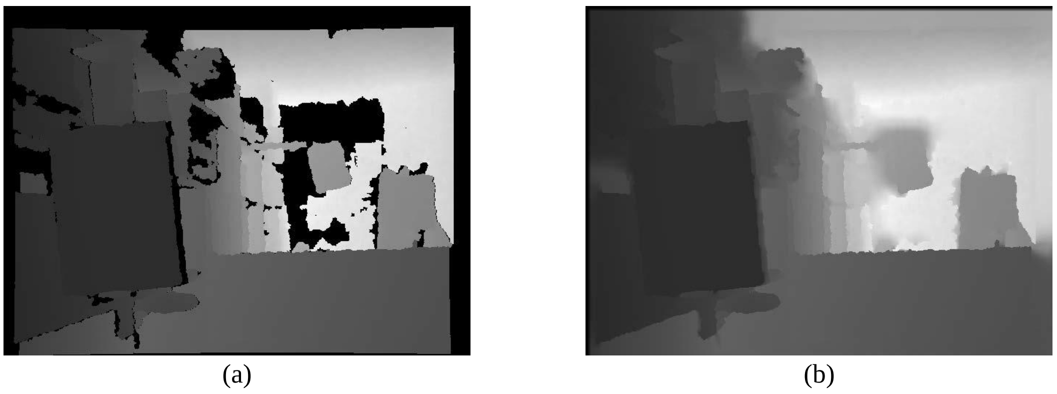


Fig. 4 Example of inpainting result, (a) Original depth raw data and (b) Inpainting results

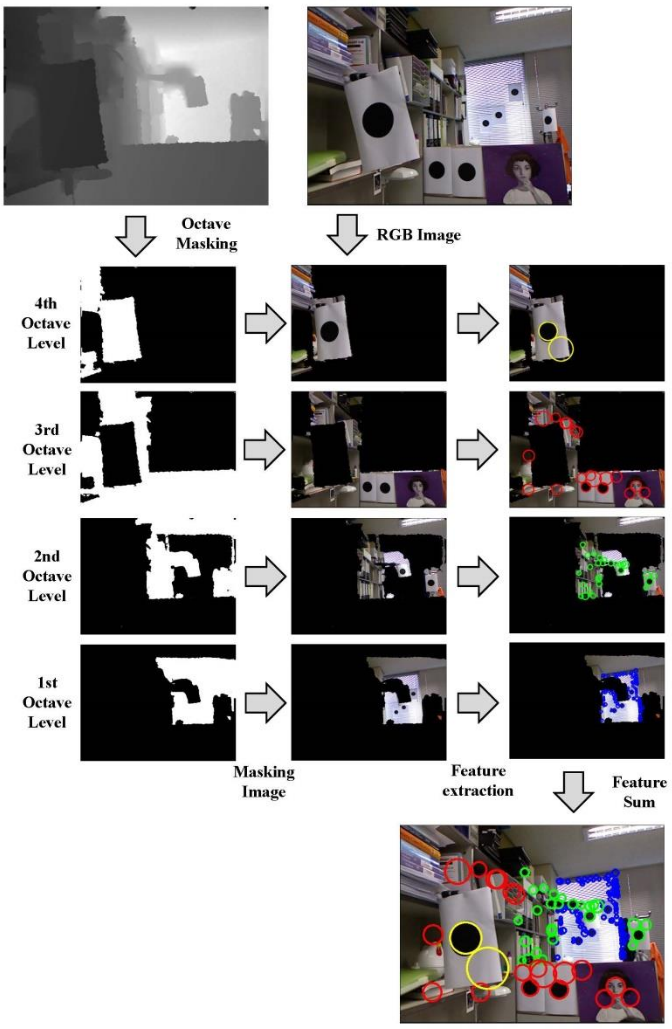


Fig. 5 Feature extraction process of DH-SURF using octave masks. The blue, green, red, and yellow circles denote extracted features from the 1st, 2nd, 3rd, and 4th octave levels, respectively

3. RGB-D SLAM system

In this study, the proposed method is validated through a graph-based RGB-D SLAM system. RGB-D sensors such as Microsoft Kinect provide depth data as well as color images (Zhang 2012). The processing steps required by the system are represented in Fig. 6. First, the 2D image features are extracted using SURF or the proposed DH-SURF method. The image features can be located at points in the 3D coordinate space using the depth data and the focal length information from the RGB-D sensor. These 3D features are used for visual odometry estimation based on comparisons between the current and preceding frames using feature matching and a random sample consensus (RANSAC) algorithm (Fischler and Bolles 1981). Next, the robot's dead-reckoning data is fused with the visual odometry result to predict robot poses. A feature manager gathers the overall features from the image frames and, based on matching between the current and previous features, the current robot pose is constrained to the past pose. This procedure is called loop closure detection. The proposed DH-SURF algorithm performs well for loop closure detection as only proper octaves are considered. However, DH-SURF algorithm occasionally incurs reduced matching results in visual odometry as only a few features are extracted by applying higher masks in far distances. Therefore, the only first octave feature is utilized in every moment for visual odometry while DH-SURF algorithm is intermittently utilized for a loop closure detection. The system is based on graph SLAM, and the robot poses and the measurement constraints are represented as a graph structure. The nodes represent robot poses while the edges constrain the nodes based on the relative measurements (the odometry prediction or the loop closure) between pairs of nodes (Fig. 7). After optimizing the graph structure using the graph SLAM algorithm such as iSAM (Kaess *et al.* 2008), a corrected robot trajectory and a 3D point cloud map can be obtained. The steps are all performed in real-time. A detailed explanation of this system is given in Lee *et al.* (2012a, b), Lee and Myung (2014).

In graph SLAM, more constraints between nodes are likely to give more reliable results of localization and mapping. In this RGB-D SLAM system, the performance of the feature extraction algorithm affects the number of the measurement constraints. Therefore, it will be shown that the proposed algorithm outperforms the classical SURF method from comparison of the results of the RGB-D SLAM experiments.

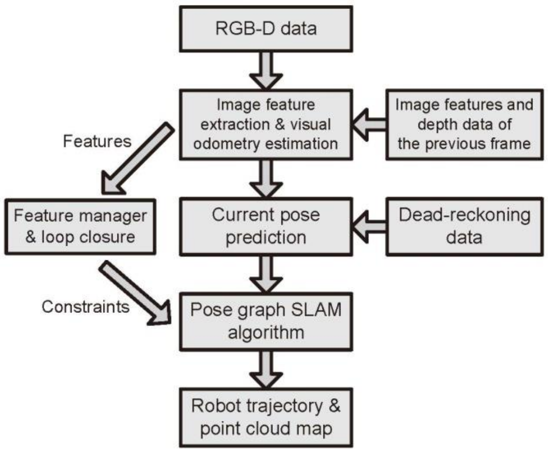

RGB-D data
Image feature extraction & visual odometry estimation
Image features and depth data of the previous frame
Features
Feature manager & loop closure
Current pose prediction
Dead-reckoning data
Constraints
Pose graph SLAM algorithm
Robot trajectory & point cloud map

Fig. 6 Processing steps required by the RGB-D SLAM system

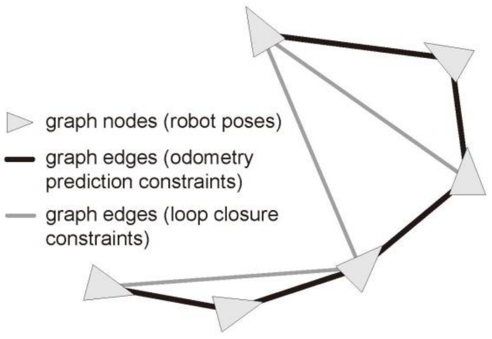
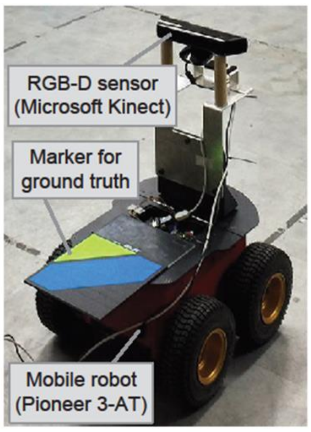

graph nodes (robot poses)
graph edges (odometry prediction constraints)
graph edges (loop closure constraints)

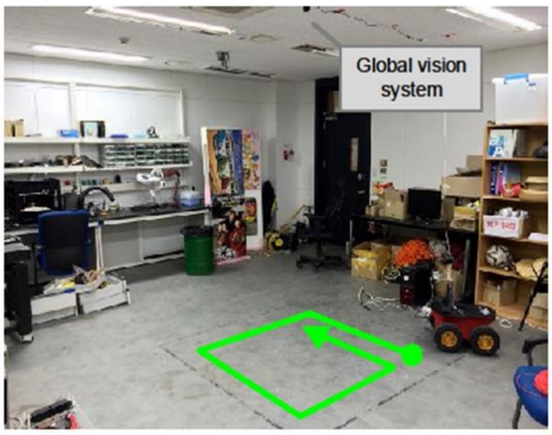
Fig. 7 Graphical model of graph SLAM

4. Experimental results

In order to demonstrate the performance of the proposed DH-SURF method, we conducted SLAM experiments with a mobile robot and a Microsoft Kinect RGB-D sensor. The Pioneer 3-AT model was used as the mobile robot (Pioneer3-AT 2015). In Fig. 8(a), the robot system equipped with an RGB-D sensor is shown along with a color marker for the ground truth position estimation. Dead-reckoning data of the mobile robot were obtained from wheel odometry that was based on 100 tick encoders. In order to measure the ground truth position, a global vision system was installed on the ceiling and a 3-degree-of-freedom (DOF) robot pose (x, y, θ) was obtained using a marker image processing. The global vision system covered 4.4×3.3 m area, and therefore, the resolution of the system was about 0.7 cm per pixel. The Kinect sensor outputs 2D RGB color images and per-pixel depth data at 30 fps both at 640×480-pixel resolution. The sensor uses structured light for depth information, and its valid range is approximately 0.4-5 m (Zhang 2012). The experiments were performed in a laboratory environment, and the robot moved along a rectangular path, as shown in Fig. 8(b).

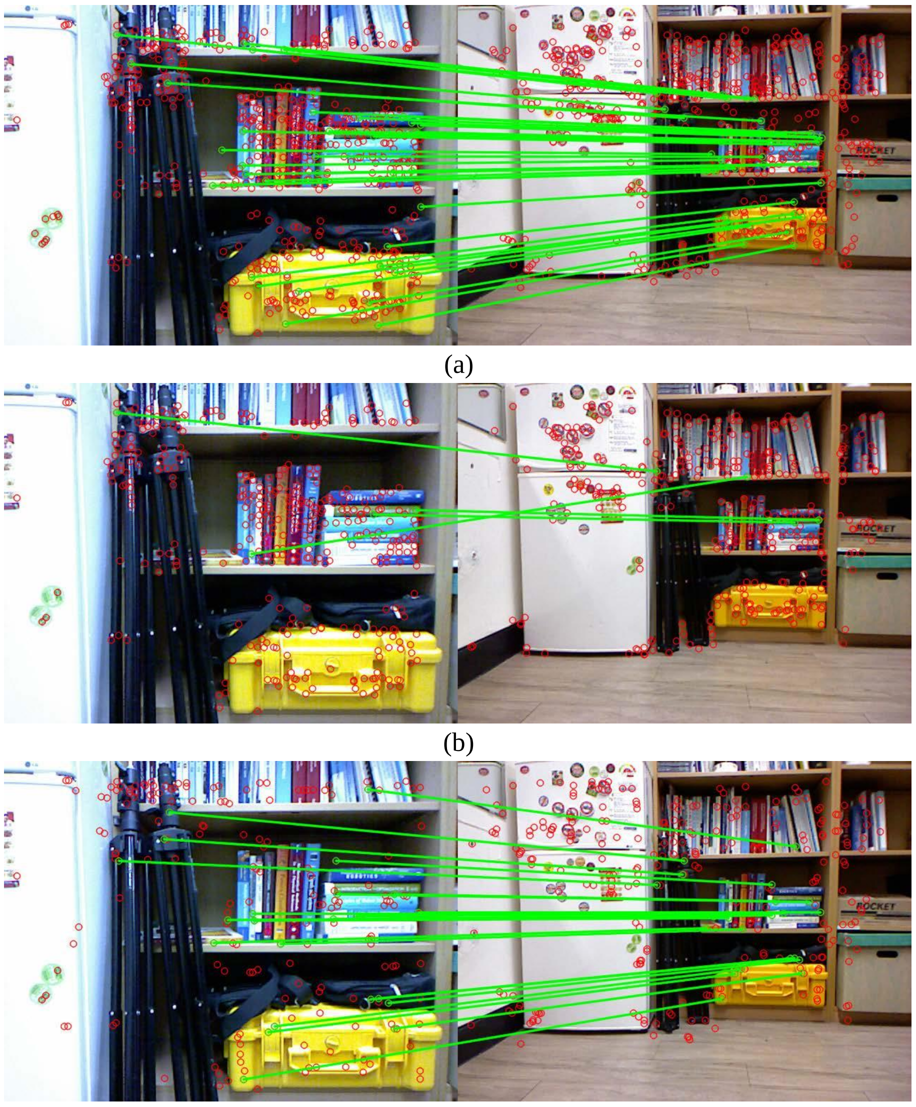


(a)



(b)

Fig. 8 (a) Mobile robot system with an RGB-D sensor and a marker for measuring the ground truth position and (b) Laboratory environment for RGB-D SLAM experiments The mobile robot moved along the rectangular path


(a)

(b)

(c)

Fig. 9 Feature matching results, (a) Classical SURF (using four octaves), (b) Classical SURF (using one octave) and (c) DH-SURF

Distance parameters d_s of DH-SURF for the octave masks are determined considering the sensing range as follows: d_4 = 0.25 m, d_3= 0.5 m, d_2 = 1.0 m, d_1 = 2.0 m, and d_0 = 5.0 m, for example. According to the previous equation, $d_s = 2 \cdot d_{s+1}$, and d_0 should have been 4.0 m. Since the Kinect sensor provides valid depth data up to 5.0 m, if the distance was larger than 4.0 m, we selected the closest condition, and finally d_0 was set to 5.0 m. According to Khoshelham and Elberink (2012), the depth data of Kinect sensor exhibit noise characteristics approximately described by standard deviations. Therefore, the standard deviations of depth error at the distance parameters d_s were set t σ_{d_4}= 1.0 cm, σ_{d_3}= 2.0 cm, σ_{d_2}= 3.0 cm, and σ_{d_1}= 7.0 cm. The DH-SURF

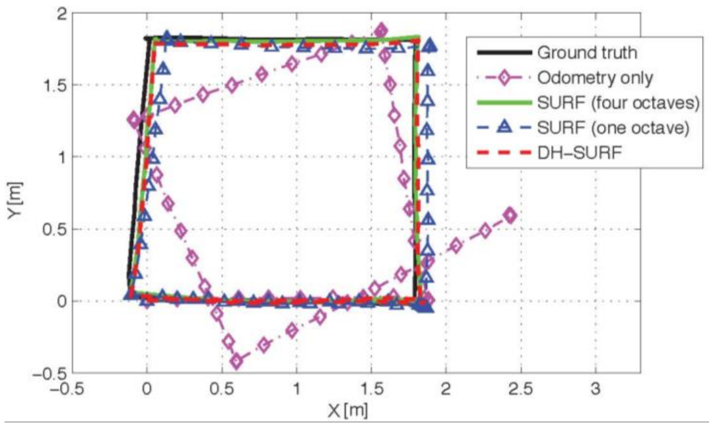


Fig. 10 Robot trajectories obtained from the RGB-D SLAM experiments

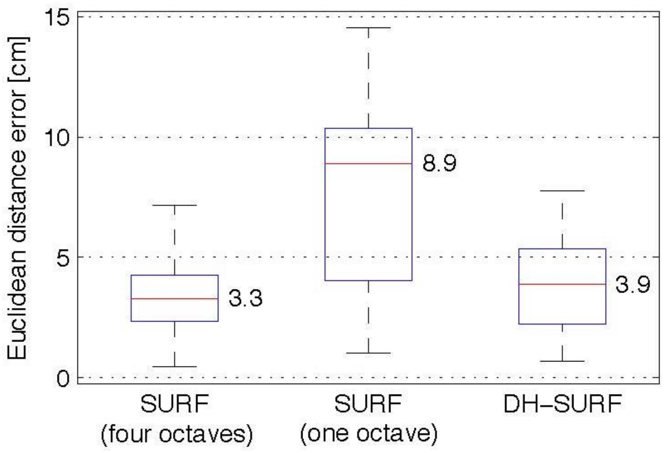


Fig. 11 Euclidean distance errors relative to the ground truth data

Table 1 Comparison of performance of the algorithms

	(a)	(b)	(c)	(d)	(e)
Classical SURF (four octaves)	348	16	1066.38	102.85	130
Classical SURF (one octaves)	147	11	476.16	82.12	108
DH-SURF	151	11	482.11	83.31	122

(a) Average processing time of feature extraction on a CPU (milliseconds), (b) Average processing time of feature extraction on a GPU (milliseconds), (c) Total processing time of the RGB-D SLAM system on a CPU (seconds), (d) Total processing time of the RGB-D SLAM system on a GPU (seconds) and (e) The number of loop closure constraints from the feature manager

algorithm was implemented on an Intel Core i5 CPU with 8 GB of memory. To accelerate the computation with a GPU, an NVIDIA GT 650 Ti graphic card supporting CUDA (compute unified device architecture) language was used.

We evaluated the performance of DH-SURF based on the graph-based real-time RGB-D

SLAM. Image features were extracted and described by the classical SURF and DH-SURF algorithms for comparisons. The algorithms were implemented both on a CPU and a GPU. Tables 1a and b show the average processing times. The processing time of classical SURF with one octave was less than that with four octaves on both processing units. DH-SURF uses one or two octaves, and hence, the processing time was similar to that of classical SURF with one octave. In performing SLAM, 58 nodes for the graph structure were made while the robot moved about 10 m along a rectangular path. And, for the loop closure detection, feature matching was performed between the images of the nodes. Fig. 9 presents an example of feature matching. The two images were at different distances, and hence, a scale-invariance property was necessary. The classical SURF with four octaves successfully found the correspondence. With only one octave, the classical SURF obtained few matching keypoints. However, the proposed DH-SURF algorithm was able to match corresponding points, even with only one or two octaves, and a constraint was successfully made. And, the total processing times of the RGB-D SLAM system on a CPU and a GPU for traversing the trajectory are shown in Tables 1c and d, respectively. The RGB-D SLAM system using the proposed DH-SURF algorithm spent less time than the classical SURF with four octaves. However, the processing time of the proposed method was similar to the classical SURF with one octave. Moreover, the RGB-D SLAM system using classical SURF with one octave ran at about 2.5 Hz and 20 Hz, on a CPU and a GPU, respectively. In the results using the proposed DH-SURF algorithm and classical SURF with four octaves, the processing rate was about 5.5 Hz and 24 Hz, on a CPU and a GPU, respectively. Table 1e shows the number of edges from the loop closure procedure in this experiment. DH-SURF made more constraints than classical SURF with one octave, thus it can be seen that the graph structure obtained from the proposed method gives better results than the graph from classical SURF with one octave.

After optimizing the graphs, corrected robot trajectories were obtained, as shown in Fig. 10. Since the experiment using DH-SURF had more constraints, the results obtained using the proposed method agree well with the ground truth than the classical SURF method. The Euclidean distance errors of the three results were calculated relative to the ground truth data, as shown in Fig. 11. The median values with classical SURF with four octaves and one octave were 3.3 cm and 8.9 cm, respectively. With the DH-SURF method, the median value was 3.9 cm. Thus, the median values obtained with classical SURF with four octaves and the proposed method had little difference.

5. Conclusions

In this paper, we have proposed a novel modified SURF algorithm called DH-SURF that uses per-pixel depth data. The proposed method maintains the scale-invariance property while reducing the computation time. Our approach was validated through RGB-D SLAM experiments. In conclusion, the proposed method performed better despite having similar processing time to that of classical SURF with one octave. As a future work, we will also adapt this algorithm to unmanned aerial robot systems for autonomous flight using RGB-D SLAM.

Acknowledgements

This research was supported by a grant (16SCIP-C116873-01) from construction technology research program funded by Ministry of Land, Infrastructure and Transport (MOLIT) of Korean government, and also supported by the Industrial Convergence Core Technology Development Program(No. 10063172, Development of robot intelligence technology for mobility with learning capability toward robust and seamless indoor and outdoor autonomous navigation) funded by the Ministry of Trade, industry & Energy (MOTIE), Korea. The students were financially supported by MOLIT through U-City Master and Doctor Course Grant Program.

References

Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. (2008), "Speeded-up robust features (SURF)", *Comput. Vis. Image Und.*, **110**(3), 346-359.

Chen, X. and Jia, Y. (2014), "Indoor localization for mobile robots using lampshade corners as landmarks: Visual system calibration, feature extraction and experiments", *J. Control Autom.*, **12**(6), 1313-1322.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D. and Burgard, W. (2012), "An evaluation of the RGB-D SLAM system", *Proceeding of the 2012 IEEE International Conference on Robotics and Automation (ICRA),* St. Paul, Minnesota, U.S.A., May.

Fischler, M.A. and Bolles, R.C. (1981), "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Commun. ACM,* **24**(6), 381-395.

Havangi, R., Nekoui, M.A. and Teshnehlab, M. (2014), "An optimization based method for simultaneous localization and mapping", *J. Control Autom.*, **12**(4), 823-832.

Henry, P., Krainin, M., Herbst, E., Ren, X. and Fox, D. (2012), "RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments", *J. Robot. Res.*, **31**(5), 647-663.

Kaess, M., Ranganathan, A. and Dellaert, F. (2008), "iSAM: Incremental smoothing and mapping", *IEEE T. Robot.*, **24**(6), 1365-1378.

Khoshelham, K. and Elberink, S.O. (2012), "Accuracy and resolution of kinect depth data for indoor mapping applications", *Sensors,* **12**(2), 1437-1454.

Kim, H., Lee, D., Oh, T., Lee, S.W., Choe, Y. and Myung, H. (2013), "Feature based 6-DoF camera localization using prior point cloud and images", *Proceeding of the 2nd International Conference on Robot Intelligence Technology and Applications (RiTA),* Denver, Colorado, U.S.A., December.

Kim, H., Oh, T., Lee, D. and Myung, H. (2014), "Image-based localization using prior map database and Monte Carlo localization", *Proceeding of the 11th Ubiquitous Robots and Ambient Intelligence (URAI),* Kuala Lumpur, Malaysia, November.

Lee, D. and Myung, H. (2014), "Solution to the SLAM problem in low dynamic environments using a pose graph and an RGB-D sensor", *Sensors*, **14**(7), 12467-12496.

Lee, D., Kim, H. and Myung, H. (2012a), "2D image feature-based real-time RGB-D 3D SLAM", *Proceeding of the 1st International Conference on Robot Intelligence Technology and Applications (RiTA),* Gwangju, Korea, December.

Lee, D., Kim, H. and Myung, H. (2012b), "GPU-based real-time RGB-D 3D SLAM", *Proceeding of the 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI),* Daejeon, Korea, November.

Lee, S. and Oh, P.Y. (2015), "Sensor information analysis for a humanoid robot", *J. Control Autom.*, **13**(1), 175-181.

Lowe, D.G. (2004), "Distinctive image features from scale-invariant keypoints", *J. Comput. Vision,* **60**(2), 91-110.

Park, S. and Park, S.K. (2014), "Global localization for mobile robots using reference scan matching", *J. Control Autom.*, **12**(1), 156-168.

Pioneer3-AT (2015), <http://www.mobilerobots.com/ResearchRobots/P3AT.aspx>.

Scaramuzza, D. and Fraundorfer, F. (2011), "Visual odometry [Tutorial]", *IEEE Robot. Autom. Magazine,* **18**(4), 80-92.

Sethian, J.A. (1996), “A fast marching level set method for monotonically advancing fronts”, *Pro. Nat. Acad. Sci.*, **93**(4), 1591-1595.

Szeliski, R. (2010*), Computer Vision: Algorithms and Applications*, Springer-Verlag New York, New York, U.S.A.

Telea, A. (2004), “An image inpainting technique based on the fast marching method”, *J. Graph. Tools,* **9**(1), 23-34.

Zhang, Z. (2012), “Microsoft kinect sensor and its effect”, *IEEE Multimedia*, **19**(2), 4-10.

HM