Fast Line Description for Line-based SLAM

Keisuke Hirose
hikeisuke@hvrl.ics.keio.ac.jp
Hideo Saito
http://hvrl.ics.keio.ac.jp/saito/

Graduate School of Science and Technology,
Keio University

Abstract

Simultaneous localization and mapping (SLAM) is a technique to simultaneously perform mapping of environments and localization of a camera in real-time. Most existing monocular vision based SLAM techniques use point features as landmarks. However, images of artificial environments with little texture often contain many line segments, whereas few point features can be localized in such a scene. We propose here a real-time line-based SLAM system, and a novel method for describing the features of line segments (LEHF:Line-based Eight-directional Histogram Feature) in order to establish correct 2D and 3D line correspondences (2D-3D correspondences). LEHF is a fast and efficient way of describing features of line segments, which are detected by the line segment detector (LSD) method. The line-based orthogonal iteration (LBOI) method and the RANSAC algorithm are applied for the camera pose estimation. We conducted an experiment in order to test our SLAM system in a desktop environment and to perform augmented reality (AR). Moreover our SLAM system was evaluated by synthetic data.

1 Introduction

Simultaneous localization and mapping (SLAM) is a technology that involves estimating both a camera pose and the structure of the environment in real-time. Vision based SLAM is used for real applications such as augmented reality[3, 6]. Most existing monocular vision based SLAM techniques employ point features as landmarks. Eade and Drummond used a FastSLAM algorithm[15] for their SLAM system with a Rao-Blackwellized particle filter that can handle a large number of landmarks[7]. Klein and Murray proposed parallel tracking and mapping (PTAM) that achieved a real-time application with several thousand landmarks[12].

Our approach uses line segments rather than points as landmarks, since there are some advantages in using line segments. Images of artificial environments with little texture contain many line segments, whereas few point features can be localized in such a scene. Moreover, line segment detection is more reliable than point detection. Line segment matching is also more robust than point matching with respect to partial occlusion and view-point changes.

This paper proposes a real-time line-based SLAM system that uses a Line-based Eight-directional Histogram Feature (LEHF), which is our new line descriptor, to establish correct 2D-3D correspondences. Existing line-based SLAM systems simply establish 2D-3D correspondences by finding the detected 2D line segment that is near the line segment (3D)

http://dx.doi.org/10.5244/C.26.83

projected by a previous camera pose in the image space. This often results in wrong correspondences being detected. With our approach, it is possible to find a detected 2D line segment that correctly corresponds to the projected 3D line segment by computing the distance between LEHFs. Hence, LEHF is used to achieve correct matching between 2D and 3D line segments and is also used for tracking line segments in mapping.

Our SLAM system was tested in a desktop environment for augmented reality (AR) and was compared with the existing approach by using synthetic data.

2 Related work

Chandraker *et al*. used stereo images for their real-time SLAM system using lines[2]. Lines of two stereo pairs (four images) were matched to compute the camera pose. Elqursh and Elgammal presented a method to estimate a relative camera pose between two images from lines[8]. Their method requires only three lines, with two of the lines parallel and orthogonal to the third.

With regard to the real-time monocular line-based SLAM, Gee and Mayol-Cuevas demonstrated a real-time SLAM system using line segments[10]. They used a modified version of Gates ’ contour tracing algorithm[9] for line segment detection and used a virtual visual servoing (VVS) method[4] to estimate a camera pose. Furthermore, the unscented Kalman filter (UKF) initializes new 3D line segments and estimates their depth. Smith *et al*. also demonstrated a real-time line-based SLAM that extended the point-based extended Kalman filter (EKF) SLAM system[5] to line correspondences[17].

These two SLAM systems[10, 17] do not use any descriptions of line segments. In their systems, the projected 3D line segment simply corresponds to the nearest detected 2D line segment in the image space to establish 2D-3D correspondences. Therefore, wrong correspondences often occur in complicated scenes that include many line segments. Zhang and Koch presented a real-time line-based SLAM that uses EKF for camera pose estimation and a line-based recovery method using angle histograms[20]. In their method, the mean-standard deviation line descriptor (MSLD)[18] that uses mean and standard deviations for the descriptions of line segments is used for matching line segments between the current frame and stored key-frames in order to relocalize the system. Since the MSLD is quite computationally expensive, they used the MSLD only for recovering. During normal SLAM procedure, nearby lines in parameter space are used for tracking.

We propose here a real-time line-based SLAM system, and a fast line descriptor (LEHF). LEHF is a fast and efficient way of describing features of line segments, which is used to establish correct 2D-3D correspondences and to track detected line segments in mapping. Also the use of LEHF provides robustness for the camera lost. We based the development of LEHF on the MSLD[18], which uses a SIFT[13]-like strategy.

3 SLAM based on line descriptors

3.1 Overview of system

In this section, we describe the overview of our system shown in Fig. 1.

In localization, LSD detects line segments and LEHF is computed for each detected line segment at every frame. In order to estimate a camera pose, 2D-3D correspondences are

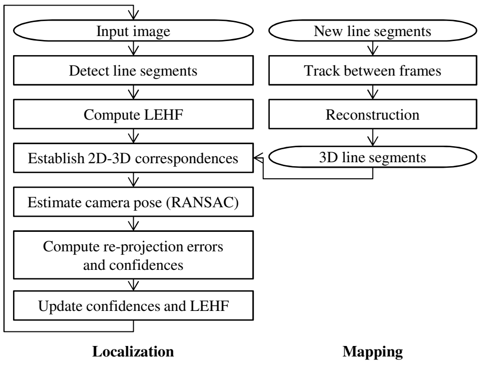

Input image
Detect line segments
Compute LEHF
Establish 2D-3D correspondences
Estimate camera pose (RANSAC)
Compute re-projection errors and confidences
Update confidences and LEHF
New line segments
Track between frames
Reconstruction
3D line segments
Localization
Mapping

Figure 1: Overview of system

established by LEHF matching. LBOI and the RANSAC algorithm estimate the camera pose from 2D-3D correspondences. Then all 3D line segments are re-projected by the estimated camera pose to compute re-projection errors and confidences which represent the reliability of 3D line segments. This confidence is used for the subsequent camera pose estimation (LBOI). In mapping, line segments are tracked between frames by using LEHF to map 3D line segments.

Before starting the system, SLAM needs to initialize the world coordinate that is the basic plane for AR. Lines in two images do not put any constraints on the camera pose[19]. Therefore, several methods to estimate the relative camera pose between two images from lines were presented[1, 8, 16]. In our SLAM system, we employ a simple way to carry out map initialization by using a fiducial marker that is detected automatically.

3.2 Localization

In this section, we describe the details of localization in which a camera pose is estimated from 2D-3D correspondences at every frame.

3.2.1 Line segment detection

We use the LSD method[11] for the line segment detection at every frame. LSD detects line segments through the gradient image with a false detection control. LSD automatically detects correct line segments even in complicated scenes that include many line segments.

3.2.2 Computation of LEHF

To achieve correct matches and make our system more robust to camera pose estimation, we compute LEHF for each detected line segment.

We based the development of LEHF on the MSLD[18], which uses a SIFT-like strategy. Since the MSLD is quite computationally expensive, it is difficult to use the MSLD in the real-time system. Therefore our line descriptor LEHF is computed fast in order to be used in the real-time system by taking a constant number of points around the line segment to compute differential values. From the computed differential values, we make eight-directional

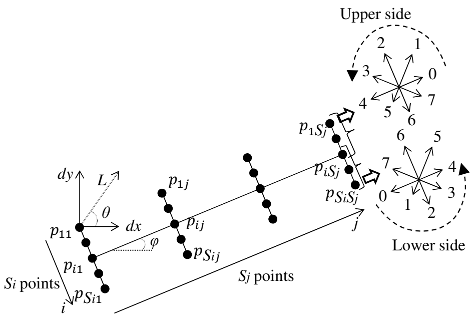


Figure 2: Overview of LEHF

gradient histograms to describe the line segment, referred to as LEHF. Since rotating the input image takes computation time, we do not rotate the input image but rectify the direction of the gradient vector computed from differential values in order to obtain rotation invariant, which is described later in detail.

Fig. 2 shows how LEHF is computed. Here, S_j is the number of points along the j axis, and S_i is the number of points along the i axis that is perpendicular to the line segment. Then we denote point $p_{ij}(px_{ij}, py_{ij})$. The dx_{ij} and dy_{ij} are the differential values for x and y directions. From dx_{ij}, dy_{ij}, a gradient vector of which the length is L_{ij} and the direction is θ_{ij} is computed. We denote the intensity value in the input image $I(px_{ij}, py_{ij})$ and $dx_{ij}, dy_{ij}, L_{ij}, \theta_{ij}$ are computed as shown in eq. 1,2,3,4. ϕ in the figure is the angle of the line segment.

$$dx_{ij} = I(px_{ij}+1, py_{ij}) - I(px_{ij}-1, py_{ij}) \tag{1}$$

$$dy_{ij} = I(px_{ij}, py_{ij}+1) - I(px_{ij}, py_{ij}-1) \tag{2}$$

$$L_{ij} = \sqrt{dx_{ij}^2 + dy_{ij}^2} \tag{3}$$

$$\theta_{ij} = \arctan(\frac{dy_{ij}}{dx_{ij}}) \tag{4}$$

As shown in the figure, we get one eight-directional gradient histogram for each i by summing S_j gradient vectors, which is denoted as $\mathbf{h_i} = (h_{i,0}, h_{i,1}, \cdots, h_{i,6}, h_{i,7})$. LEHF is obtained by merging all $\mathbf{h_i}$, which is denoted as $\mathbf{d} = (\mathbf{h_1}, \mathbf{h_2}, \cdots, \mathbf{h_i}, \cdots, \mathbf{h_{(S_i+1)}})$. Why we compute $(S_i + 1)$ histograms is that two histograms are computed for the center point (on the line segment).

However if we simply merge all $\mathbf{h_i}$, computed LEHFs are not matched between the images that one image is rotated 180 degrees since the directions of $\mathbf{h_i}$ are not matched. Therefore, LEHF is designed symmetrically. As shown in the figure, the directions of $\mathbf{h_i}$ of lower side is inverted to that of upper side. Moreover, computed $\mathbf{h_i}$ are merged symmetrically to obtain $\mathbf{d}$. Since both histograms (upper side and lower side) are computed for the center point (on the line segment), we get $(S_i + 1)$ histograms.

If we assume that $S_i = 5$, 6 eight-directional gradient histograms are obtained and we compute $\mathbf{d}$ symmetrically as eq. 5. However, as LEHF is computed symmetrically, we need to compute another distance between LEHFs that one LEHF vector is inverted.

$$\mathbf{d} = (h_{1,0}, \cdots, h_{1,7}, h_{2,0}, \cdots, h_{2,7}, h_{3,0}, \cdots, h_{3,7}, h_{4,7}, \cdots, h_{4,0}, h_{5,7}, \cdots, h_{5,0}, h_{6,7}, \cdots, h_{6,0}) \quad (5)$$

The distance between p_{ij} and $p_{i+1,j}$ is three pixels. The weight for each i is $w_i = G_\sigma((i - \frac{S_i+1}{2})3)$. G_σ is a normal distribution that has the variance σ. The argument of G_σ is the distance from the center point. The algorithm for computing LEHF is as follows.

1. Firstly, initialize LEHF $(\mathbf{h_1}, \cdots, \mathbf{h_{(S_i+1)}})$ by zero.
2. For point p_{ij}, dx_{ij} and dy_{ij} are computed.
3. A gradient vector that has lengthL_{ij} and directionθ_{ij} is computed from dx_{ij} and dy_{ij}.
4. θ_{ij} is rectified by subtracting ϕ from θ_{ij} to obtain rotation invariant.
5. θ_{ij} is quantized into eight directions to obtain the bin($id = [0 \sim 7]$) of the $\mathbf{h_i}$.
6. $\mathbf{h_i}$ is updated by $h_{i,id} \leftarrow h_{i,id} + w_i \times L_{ij}$.
7. For all points $(p_{11} \sim p_{S_iS_j})$, step $2 \sim 6$ are carried out.
8. Merge $\mathbf{h_1}, \cdots, \mathbf{h_{(S_i+1)}}$ symmetrically (eq. 5) to obtain $\mathbf{d}$ and normalize it to set the norm of $\mathbf{d}$ one.

We used 45 points for S_j and 13 points for S_i in the experiments. Therefore, we get an $8 \times (13+1) = 112$ dimensional descriptor for the LEHF.

3.2.3 Establishment of 2D-3D correspondences

2D-3D correspondences are established by LEHF matching to estimate the camera pose. In our system, each 3D line segment has one state of three which is calculated by re-projection errors. First one denotes an inlier line. Second one denotes an outlier line. Whether the line is inlier or outlier is simply calculated by a threshold of the re-projection error that is also used in RANSAC. Third one denotes that the line is outside of the image space. In our system, only 3D line segments that have the state of inlier are projected by a previous camera pose. Then, for each projected 3D line segment, we search some detected 2D line segments which are near from the projected 3D line segment in the image space and the angular difference with the projected 3D line segment is within the threshold. LEHF distance is computed for those searched 2D line segments and finally the 3D line segment corresponds to a 2D line segment that has the minimum Euclidean distance between LEHFs.

3.2.4 Camera pose estimation

In order to estimate a camera pose robustly, we use the RANSAC algorithm and the LBOI method[21] that is the extended method of OI algorithm[14] to lines. Since the LBOI method can estimate the camera pose from at least three pairs, three pairs of a 2D-3D correspondence are randomly chosen and the camera pose is estimated in the iterations of RANSAC. For each estimated camera pose, re-projection errors are computed for all 2D-3D correspondences. The distance and the angular difference between the detected 2D line segment and the re-projected 3D line segment are computed as the re-projection error, which are shown in the Fig. 3. To reduce computational cost, we simply take a center point of l_1 and consider the line orthogonal to l_1. Then E_{dist} is the distance between the center point of l_1 and the intersection point between l_2 and the orthogonal line. E_{angle} is the angular difference between l_1 and l_2. Then the re-projection error E is obtained by summing E_{dist} and E_{angle}. If E is smaller than the threshold E_{th}, the 2D-3D correspondence is inlier. If E is larger than E_{th}, the 2D-3D correspondence is outlier. By updating the maximum inliers in the iterations of RANSAC, we select the set of maximum inliers of 2D-3D correspondences. Then we estimate the final camera pose from those maximum inliers of 2D-3D correspondences by the LBOI. Selected 2D-3D correspondences are weighted by the confidence of each 3D line segment in the final LBOI estimation.

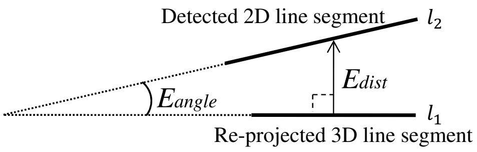


Figure 3: E_{dist} and E_{angle} are computed for a pair of 3D and 2D line segments(l_1 and l_2).

3.2.5 Updating of confidences and LEHF

In our system, each 3D line segment has LEHF and a confidence representing the reliability of the 3D line segment. The confidence that has a value of $0 \sim 1$ is computed by the re-projection error. Confidences and LEHF of 3D line segments are updated at every frame.

We re-project all 3D line segments by the estimated camera pose to compute re-projection errors. The re-projection error E is computed in the same way as the camera pose estimation 3.2.4. Here, the corresponding 2D line segment is the nearest line segment from the re-projected 3D line segment in the images space. As I described in sec. 3.2.3, each 3D line segment has one state of three. If E is smaller than E_{th}, the state of the 3D line segment will be inlier. If E is larger, the state will be outlier. If the re-projected line segment is outside of the image space, the state will be outside of the image. For the outside 3D line segment, nothing is done. For the inlier 3D line segment, the new confidence is computed and the confidence is updated by averaging the current confidence and the new confidence. We set the converted confidence value of the re-projection error E_{th} 0.8 and the new confidence C_{new} is simply computed as $C_{new} = 1-(E/(E_{th} \times 5))$. Why we multiply E_{th} by 5 is to set the converted confidence value of E_{th} 0.8. The LEHF is also updated with that of corresponding 2D line segment. For the outlier 3D line segment, the current confidence is decreased. The 3D line segment of which the confidence is lower than zero is removed from the system. E_{th} is set 5 in experiments.

In our system, newly mapped 3D line segments are not used for localization during the first 20 frames. Newly mapped 3D line segments that are re-projected into outside of the image space are immediately removed.

3.3 Mapping

Mapping starts regularly. In mapping, line segments not used for localization are tracked between frames as new line segments. Computed LEHF is used to track line segments correctly. A 3D line that is estimated from two images is a line of intersection between two planes in the Fig. 4. We track 8 images and establish 7 pairs of two images as image 1-2, image 1-3, $\cdots$, image 1-8. Then a 3D line is estimated for each pair and 7 3D lines are computed. A final 3D line is estimated by averaging those 7 3D lines.

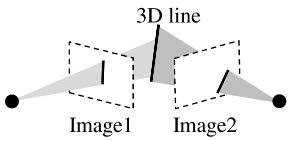
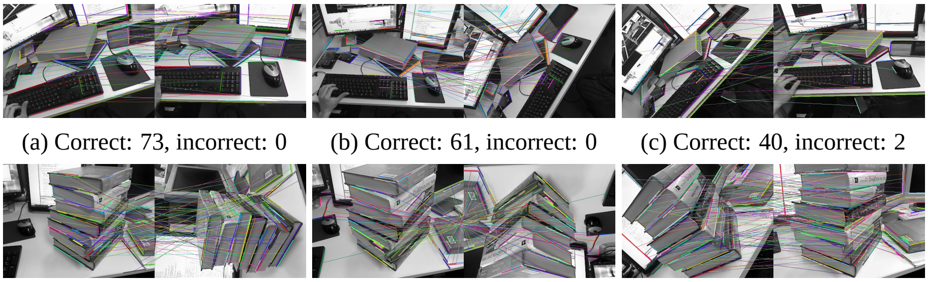


Figure 4: Estimation of a 3D line from two images.

4 Experiments

4.1 LEHF matching

Fig. 5 shows the results of LEHF matching. We manually counted the number of correct and incorrect matches. In the figure, most matches are correct even if there is some rotation between two images.



(a) Correct: 73, incorrect: 0 (b) Correct: 61, incorrect: 0 (c) Correct: 40, incorrect: 2

(d) Correct: 80, incorrect: 3 (e) Correct: 73, incorrect: 5 (f) Correct: 120, incorrect: 16

Figure 5: Results of LEHF matching.

4.2 Experimental results of SLAM system

4.2.1 Synthetic data experiments

We evaluated our SLAM system by using synthetic data. We built a simple synthetic environment that is shown in the left image of Fig. 6. Then 447 sequential images were generated with perfectly known camera poses. In this experiment, the fiducial marker of which size is $10cm \times 10cm$ was used for initialization. After 153th frame, the fiducial marker disappeared as shown in the right image of Fig. 6 and the camera pose was estimated by using only mapped 3D line segments. Moreover, we compared our SLAM system with the standard line-based SLAM which did not use LEHF for tracking in mapping and for establishing 2D-3D correspondences. The standard line-based SLAM used nearest neighbor (NN) search that searched the nearest line segment in the image space as existing line-based SLAM methods. The other methods (line segment detection, camera pose estimation, etc.) are completely same as our SLAM system. The results are shown in Fig. 7. In our system, when the number of 2D-3D correspondences are too small or the camera move distance between the estimated camera pose and the previous camera pose is too long, the system assumes that the the camera pose estimation failed in order to prevent updating by bad parameters. In 447 frames, the standard NN method failed to estimate for 65 frames whereas our SLAM system failed to estimate for only 7 frames. We calculated the root mean squared error with ground truth for each graph except failed frames and we show them in Tab. 1. As shown in the table, the results of our SLAM system using LEHF are more accurate than the standard NN method.

Figure 6: Constructed synthetic environment.

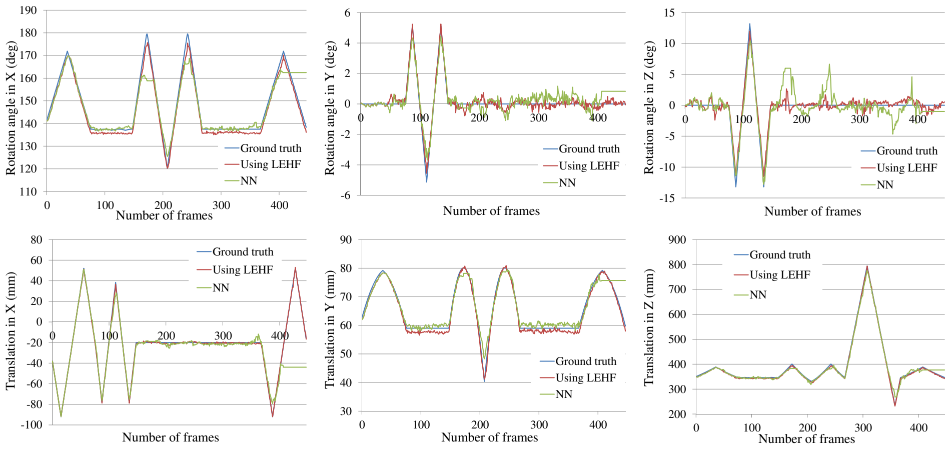


Figure 7: Comparison of the ground truth and our SLAM and the standard NN method.

	Rot. X (deg)	Rot. Y (deg)	Rot. Z (deg)
Proposed method	2.03	0.24	0.73
Standard NN method	2.53	0.43	1.40
	Trans. X (mm)	Trans. Y (mm)	Trans. Z (mm)
Proposed method	0.85	1.12	3.96
Standard NN method	2.93	1.57	7.62

Table 1: RMSE between the estimated values and ground truth.

4.2.2 Real data experiments

In this experiment, we captured 640×480 images from a web camera(Logicool Webcam Pro 9000) and tested our SLAM system on a desktop PC (CPU:Intel(R) Core(TM) i3 CPU 3.07 GHz, Memory: 2.92 GB). Fig. 8 shows mapped 3D line segments and estimated camera poses. Our SLAM starts with a fiducial marker, which is shown in the first image of Fig. 9. In Fig. 9, green line segments are inlier line segments and red line segments are outlier line segments. Some augmented reality scenes are shown in Fig. 10. The time required to process the tasks in Fig. 8 is plotted in Fig. 11. The mean processing time of 1087 frames was 87.78ms and mean fps was 11.39fps.

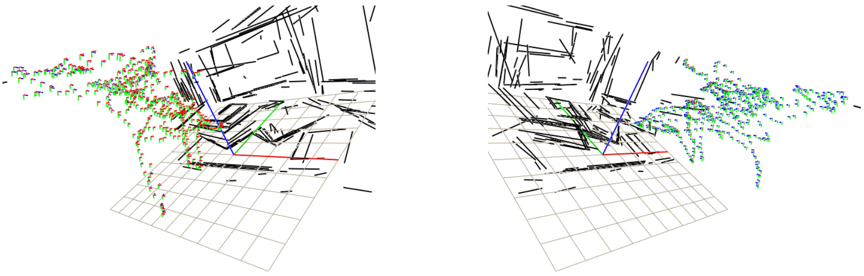


Figure 8: Mapped 3D line segments and estimated camera poses.

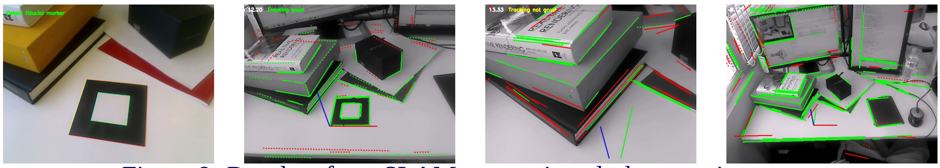


Figure 9: Results of our SLAM system in a desktop environment.

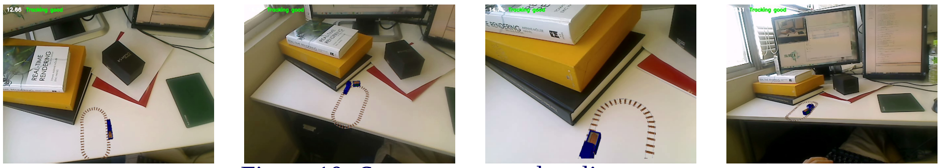


Figure 10: Some augmented reality scenes.

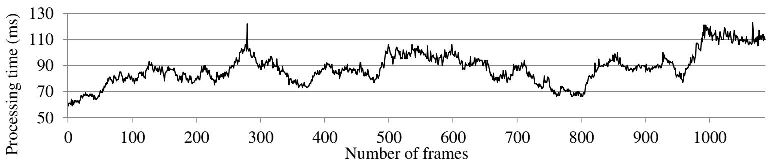


Figure 11: Processing time of our SLAM system.

5 Conclusion

We have presented a real-time line-based SLAM system that uses a line descriptor called a LEHF. We showed how our SLAM system can be used in a small desktop environment. By using LEHF, 2D-3D correspondences are established correctly and the camera poses are robustly estimated as shown in the experimental results. Also, newly detected line segments are correctly tracked between frames in mapping. The use of line segments as landmarks provides robust detection with respect to large viewpoint changes and partial occlusion. Our SLAM system was evaluated by synthetic data and demonstrated for AR in a desktop environment.

Acknowledgements

This work was partially supported by MEXT/JSPS Grant-in-Aid for Scientific Research(S) 24220004.

References

[1] H. Bay, V. Ferrari, and L.V. Gool. Wide-baseline stereo matching with line segments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 329–336, June 2005.

[2] M. Chandraker, J. Lim, and D. Kriegman. Moving in stereo: Efficient structure and motion using lines. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1741–1748. IEEE, 2009.

[3] D. Chekhlov, A. Gee, A. Calway, and W. Mayol-Cuevas. Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, November 2007.

[4] A.I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, pages 615–628, 2006.

[5] A.J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410. IEEE, 2003.

[6] A.J. Davison, W.W. Mayol, and D.W. Murray. Real-time localisation and mapping with wearable active vision. 2003.

[7] E. Eade and T. Drummond. Scalable monocular slam. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:469–476, 2006. ISSN 1063-6919.

[8] A. Elqursh and A. Elgammal. Line-based relative pose estimation. *Computer Vision and Pattern Recognition(CVPR), IEEE Conference on*, pages 3049–3056, 2011. ISSN 1063-6919.

[9] J. Gates, M. Haseyama, and H. Kitajima. A new conic section extraction approach and its applications. *IEICE - Trans. Inf. Syst.*, E88-D:239–251, February 2005. ISSN 0916-8532.

[10] A. Gee and W. Mayol-Cuevas. Real-time model-based slam using line segments. In *2nd International Symposium on Visual Computing*, 2006.

[11] R. Grompone von Gioi, J. Jakubowicz, J.M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010.

[12] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, 2007.

[13] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, November 2004. ISSN 0920-5691.

[14] C.P. Lu, G.D. Hager, and E. Mjolsness. Fast and globally convergent pose estimation from video images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(6):610–622, 2000.

[15] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using fastslam. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1985–1991. IEEE, 2003.

[16] O. Pellejero, C. Sagüés, and J. Guerrero. Automatic computation of the fundamental matrix from matched lines. *Current Topics in Artificial Intelligence*, pages 197–206, 2004.

[17] P. Smith, I. Reid, and A. Davison. Real-time monocular slam with straight lines. In *British Machine Vision Conference*, volume 1, pages 17–26, 2006.

[18] Z. Wang, F. Wu, and Z. Hu. Msld: A robust descriptor for line matching. *Pattern Recognition*, 42(5):941–953, 2009.

[19] J. Weng, T.S. Huang, and N. Ahuja. Motion and structure from line correspondences; closed-form solution, uniqueness, and optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):318–336, 1992.

[20] L. Zhang and R. Koch. Hand-held monocular slam based on line segments. In *2011 Irish Machine Vision and Image Processing Conference*, pages 7–14. IEEE, 2011.

[21] X. Zhang, K. Wang, Z. Zhang, and Q. Yu. A new line-based orthogonal iteration pose estimation algorithm. In *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, pages 1–4. IEEE, 2009.