Evidence gathering for line based recognition by real plane

이재규*, Jae-Kyu Lee, 류문욱**, Moonwook Ryu, 이장원**, Jangwon Lee

Abstract We present an approach to detect real plane for line base recognition and pose estimation. Given 3D line segments, we set up reference plane for each line pair and measure the normal distance from the end point to the reference plane. And then, normal distances are measured between remains of line endpoints and reference plane to decide whether these lines are coplanar with respect to the reference plane. After we conduct this coplanarity test, we initiate visibility test using z-buffer value to prune out ambiguous planes from reference planes. We applied this algorithm to real images, and the results are found useful for evidence fusion and probabilistic verification to assist the line based recognition as well as 3D pose estimation.

Keywords: coplanarity, evidence fusion, plane sweep, real plane, visibility.

본 논문은 2007년 성균관대학교 BK (Brain Korea) 학술 연구비 지원에 의하여 연구되었음.
*주저자 : 성균관대학교 지능시스템 연구센터, e-mail : jklee1965@skku.edu
**공동저자 : 정보통신공학부, e-mail : moonwook@ece.skku.ac.kr, jwlee31@ece.skku.ac.kr

1. Introduction

Recognizing 3D object and estimate its pose in real environment is challenging task. Because the scene from real environment dose have many noisy and uncertainty, its modeling representation and recognition become difficult. This is because the feature information from real environment is so ambiguous and uncertain that any single group of information cannot give significant contribution for recognition. One of attempts to resolve this problem is evidence fusion in probabilistic approach. Our goal is, therefore, to collect variety of evidence from color, texture, 3D line segments, and planes. Amongst them, what we are devoted to is detect real planes using 3D line segments. 3D lines belong to the same object or environment are located at the same plane in 3D space even under changes of viewing angles. However, those lines are scattered in 3D space and therefore interpret their pattern for recognition directly is very confusing procedure. Therefore classify 3D line's correspondence along with their real plane information is very helpful to reduce the ambiguity and uncertainty during the model matching process. To find such real planes, we conduct visibility test and plane sweep algorithm. As preprocess, we collect a group of 3D line segments. These 3D lines can be obtained from 2D segmentations and 3D point clouds which obtained from stereo images. Then, at run-time, we first build up reference planes using selected parallel 3D line pair. To build reference plane, we firstly build implicit plane by set up plane equation using three endpoints from given parallel line pairs. Then we measure the normal distance from that plane to the remain fourth endpoint. Then using plane sweep algorithm, we find coplanar lines which are belong to each reference plane. Threshold is defined to determine the coplanarity to the plane in boolean way. Our real plane search algorithm is based on Baillard and Zisserman [1] which is to find and sort out half planes of building roofs from areal photographs. Then we conduct visibility test to prune out ambiguous planes and classify real plane candidates using z-buffer information. We have implemented our proposed algorithm using Visual C++ language for the real as well as synthetic images, and the computation takes within 20 msec depends on number of input lines on a dual core 3 GHz Pentium 4 machine. We also present an error analysis and algorithm

complexity to quantify the amount of error induced by threshold measure and the amount of complexity created by line pair selection to build reference planes, respectively.

2. Related Works

Line based recognition and real plane construction have been extensively investigated by researchers in image processing, computer vision, and intelligent robotics area. In this section, we review those works directly related to our work. To find line match for object or environment recognition, there have been many approaches to achieve this. Zhang and Faugeras [2] applied Kalman filter approach. Another try is to trace transformation of line's endpoints by iterative closest points (ICP) Chen and Medioni [3], Besl and MaKay [4]. But this method is tracing line's endpoints by individually not by pairs. Recently, such works are improved by Kamgar-Parsi and Kamgar-Parsi [5] for matching almost equal length of line pairs. Another different approach is matching Hausdorff distance between two line segments Guerra and Pascucci [6]. In this method, they used not only Hausdorff distance but also line triplet for iterating transformation between lines. Those approaches are to apply least square sense to transformation matrix which can trace the coordinates of endpoints or midpoints and try to find their match by iterative schemes. However still their applications are limited to synthetic images and when the transformation difference is large, then they rarely converge. There are many methods to obtain real planes using from 3 views by Schmid and Zisserman [7] to 6 views Baillard and Zisserman *et al* [8]. Another approaches are detecting and regrouping neighboring coplanar 3D lines computed from the images by Bignone and *et al* [9]. As our good reference for real plane reconstruction by 3D points and 3D lines, we recommend Criminisi, Reid, and Zisserman [10], and Baillard and Zisserman [1], respectively. But most of their application is for areal photographs or outdoor buildings, and they need many sequential images and camera informations. The study on plane sweep and visibility test also has a wealthy amount of literature. Plane sweep algorithm is a type of algorithm that uses a conceptual line or surface to solve various problems in Euclidean space. The key idea is to imagine that a line or a plane is swept or moved across the plane, stopping at some points. Several algorithms have been developed that approach and achieved the optimal goal for general case to find intersection points [11] [12] [13], to compute arrangements [14] [15], and to find intersections in the red/blue case [16].

3. Preliminaries

3.1 Extracting 3D Lines

The 2D lines are obtained by applying a local implementation of the Canny edge detector, detecting discontinuities in the edge chains, and straight line estimation by regression. Then, they are mapped with 3D point clouds from stereo camera. And then we obtain 3D lines with least square sense. Fig.1 shows the procedure how to generate 3D lines from 2D edges and 3D point clouds. More details are in [17][18].

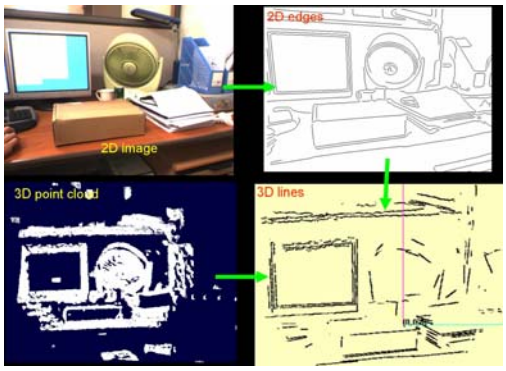

Fig. 1. 3D line extraction process and results

3.2 Build Reference Planes and Find Co-Planes

As the next step, we are building reference plane from a given 3D line segments. From 3D lines, we firstly construct 3D planes using pair of 3D lines. Using any three endpoints from line pair, plane equation can be found. Then measure the normal distance between that plane and the fourth endpoint. We call this as reference plane if the fourth endpoint is close enough to the plane. And therefore the reference plane is implicit (dimensionless) plane in terms of plane equation without any boundary.

And now we use these implicit planes to find a coplanarity of the remains of 3D line segments using

plane sweep algorithm. In geometry based image processing, 3D line is one of the most persistent and rigid data which can represent features. Therefore 3D lines belong to the same object are located in the same real plane. And our objective is to find such planes and their lines. Fig. 2 shows the coplanarity and parallelism test in graphical scheme. By conducting coplanarity test, we make those dimensionless plane as explicit plane which has dimension and vertexes. There are two coplanarity tests.

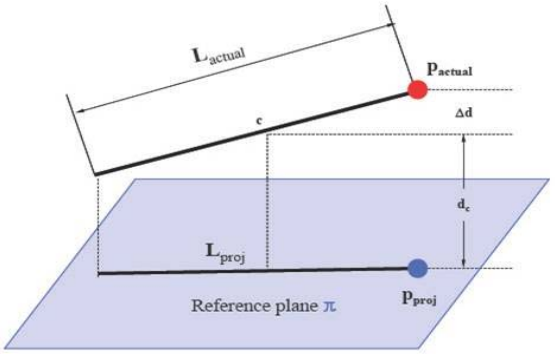

Fig. 2. Coplanarity and Parallelism test between line and reference plane.

One is for line pair when the reference plane is built. If the plane is not coplanar to the given line pair, then building process for the reference plane is ceased. The other one is for coplanar test between the reference plane and the remains of 3D line segments. For coplanarity test, we use the distance measure. The first distance measure that we tried was measure the Euclidean distance between actual line L_a and the projected line L_p as $d_c = L_a - L_p$. Because this line distance measure is absolute distance based on 3D line's endpoint, therefore, the accuracy for very short or long line cannot be properly evaluated. To remedy this problem, we also consider each line's orientation with respect to reference plane which is $atan = \frac{\triangle d}{(La/2)}$. And then, we define explicit plane by set plane's dimensions from four endpoints.

3.3 Visibility Test

The procedure for visibility test using depth information from OpenGL [19] is as follows: First, we render all 3D lines and co-planes. Then second, we read their z-buffer and mapping the information to the (u, v) coordinate of image planes. Then also find their matched (x, y, z) coordinates of line or plane per each pixel. By doing this for every pixel, we can verify whether any co-plane is front or behind any 3D line. If any co-plane is located in front of line, then it will be prune out. By this method, we prune out ambiguous planes. The co-planes which passed visibility test can be classified as real plane candidates. Fig. 3 shows visibility test scheme.

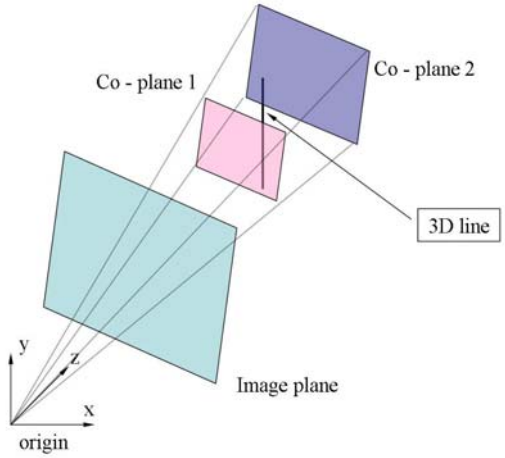

Fig. 3. Visibility Test

In Fig. 3, a 3D line is located between two rectangular shape co-planes. In this case, we have depth test and have a result that one plane is in front of line (like co-plane 1), and the other one is at rear (like co-plane 2). And if it is real plane, there is no possibility that 3D line is stand at rear. Using this clue, we prune out ambiguous planes or (ghost planes).

3.4 Algorithm Complexity and Error Analysis

Our algorithm is simple but takes $O(n^2)$ for reference plane evaluation where n is number of 3D lines. For plane sweep, however, the complexity becomes $n\log(n+k)$ where k is number of co-planes. The other issue is measurement error. Until now, we just use Euclidean distance for coplanarity evaluation as well as 3D line extraction process. However, the 3D lines

extracted from 3D point clouds are also containing many uncertain factors. Therefore, at each step, more theoretical approach is needed to remedy these errors.

4. Experimental Results

We implemented the entire pipeline of our algorithm on a PC equipped with a Intel Dual Core 3GHz Pentium Processor, 2GB of main memory. As a choice for programming languages, we used Visual C++ 6.0 and OpenGL 2.0 Libraries. Our bench marking scenario is as follows: set reference planes and define implicit surface using given parallel line pairs. Then define co-planes and apply visibility test. We implement our algorithm for real images. During the implementation, the computation time also has been measured. We have results from box image. In Fig. 4, firstly, 3D point clouds are obtained from stereo camera, and 3D lines are extracted in the next. Then, co-planes classified from reference planes. And finally, we collect real planes. The images are constructed by structured light camera for visual convenience.

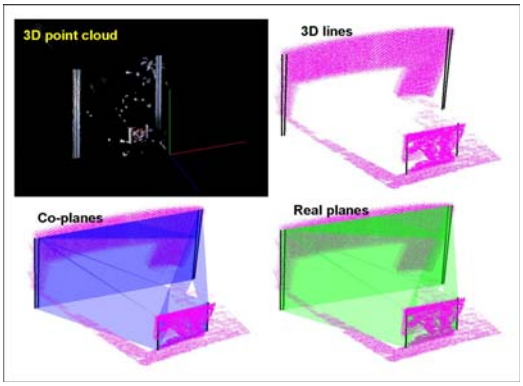

Fig. 4. Experimental Results

For this simple example, however, we have limitation on real plane classification. Without proper number and distribution of 3D lines, we are unable to eliminate all virtual planes from co-planes. The last picture shows such problem. Still many virtual planes remain. However, we can use this real plane candidates for other evidence selection for model matching. We will conduct test for more complicated images and improve our algorithm. Fig.5 shows computation time. The first one is visibility test and the second one is the remaining process including reference plane and co-plane build up. As we can see, the visibility test takes a lot of time compare to the remains. This is mainly due to checking every pixel and mapping their (u, v) coordinates to the object (x, y, z) coordinates. It is obvious that there is need to improve this procedure. One of consideration now is using (Graphic Processing Unit) GPU

5. Conclusions

We have introduced an algorithm to detect coplanar lines and their real planes for evidence fusion in line based recognition. We have performed our measurements for real image and also measure the computation time. So far, we found the limitations of our algorithm that we have to consider as follows: In line based recognition, since using only fundamental geometry information such as parallel or intersect lines are insufficient evidence, it can cause a lot of false positives. Hence those matching results can give only tentative solution for verification. Furthermore, to obtain perfect 3D lines from stereo images is still under developing area. There are many inaccurate factors during 2D segmentation as well as achieving 3D line segments from stereo images. Such limitations also must be compensated by other more efficient techniques.

References

[1] C. Baillard and A. Zisserman. "A plane-sweep strategy for the 3D reconstruction of buildings from multiple images", In Proc. 19th. ISPRS Congress and Exhibition, 2000.

[2] Z. Zhang and O. Faugeras, "Determining motion from 3D line segment matches: A comparative study" , Image and Vision Computing, 9(1): 10 ~ 19, 1991.

[3] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images", Proceeding of IEEE ICRA, pages 2724 ~ 2729, April 1991.

[4] P. Besl and N. MaKay. "A mothod for registration of 3-D shapes", IEEE PAMI, 14(2):239 ~ 256, Feb. 1992.

[5] B. Kamgar-Parsi and B. Kamgar-Parsi. "Algorithms for matching 3D line sets", IEEE PAMI, 26(5):582 ~ 593, May 2004.

[6] C. Guerra and V. Pascucci. "Line-based object recognition using Hausdorff distance: from range

images to molecular secondary structures", Image and Vision Computing, 23(4):405 ~ 415, April 2005.

[7] C. Schmid and A. Zisserman. "Automatic line matching across views", IEEE Conference on Computer Vision and Pattern Recognition, pp. 666 ~ 671, 1997.

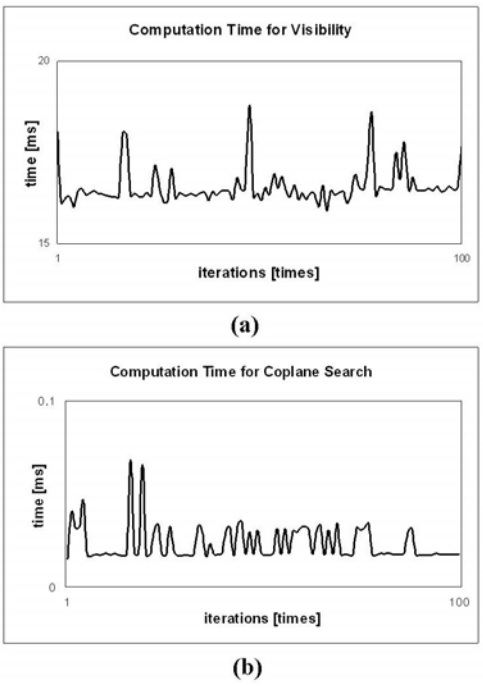

Fig. 5. Computation Time

[8] C. Baillard, A. Zisserman and et al. "Automatic line matching and 3D reconstruction of buildings from multiple views", ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagimary, IAPRS, Vol. 32, Part 3-2W5, pp. 69 ~ 80, 2000.

[9] F. Bignone and et al. "Automatic extraction of generic house roofs from high resolution aerial imagery", European Conference on Computer Vision, pp. 85 ~ 96, 1996.

[10] A. Criminisi, I. Reid and A. Zisserman. "A plane measuring device", Im Proc. BMVC, Sep. 1997.

[11] J. D. Boissonnat and J. Snoeyink. "Efficient algorithms for line and curve segment intersection using restricted predicates", In Porceeding, 15th Annual ACM Symposium of Computational Geometry, pages 370 ~ 379, 1999.

[12] I. Balaban. "An optimal algorithm for finding segment intersections", In Porceeding, 11th Annual ACM Symposium of Computational Geometry, pages 211 ~ 219, 1995.

[13] U. Bartuschka, K. Mehlhorn, and S. Naher. "A robust and efficient implementation of a sweep line algorithm for the straight line segment intersection problem", In Porceeding, Workshop on Algorithm Engineering, pages 124 ~ 135, 1997.

[14] J. L. Bentley and T. A. Ottoman. "Algorithm for reporting and counting geometric intersections", IEEE Trans. Computation, C-28(9): pages 643 ~ 647, Sep. 1979.

[15] J. D. Boissonnat and F. P. Preparata. "Robust plane sweep for intersecting segments", SIAM Journal of Computation}}, 29(5), pages 1401 ~ 1421, 2000.

[16] L. Palazzi and J. Snoeyink. "Counting and reporting red/blue segment intersections", CVGIP: Graphic models image processing, 56(4), pages 304 ~ 311, 1994.

[17] Samuel H. Chang, Sukhan Lee, Dongju Moon, Woongmyung Kim, and Yeunghak Lee. "Model based 3D Object Recognition using Line Features", International Conference on Advanced Robotics (ICAR), 2007.

[18] Sukhan Lee, Seongsoo Lee, Jeihun Lee, Dongju Moon, Eunyoung Kim and Jeonghyun Seo. "Robust Recognition and Pose Estimation of 3D Objects Based on Evidence Fusion in a Sequence of Images", Robotics and Automation, IEEE International Conference (ICRA) on 2007.

[19] http://www.opengl.org/resources/.