Self Localization with Edge Detection in 3D Space

Vaishali Ailani, Disha Prakash, and K. S. Venkatesh
Indian Institute of Technology, Kanpur, India
vaishaliailani@gmail.com, venkats@iitk.ac.in

***Abstract*— The paper describes our contribution in the field of visual localization as a desirable feature of an autonomous mobile robot in an unknown environment. Reviewing current approaches, our paper provides an improved algorithm to track 3D straight edges, facilitated by the use of Kinect (a camera developed by Microsoft corporation) to provide RGB color image as well as the depth map through IR sensor on it. Correspondence between edges in two consecutive frames is established through edge templates. Following this is the determination of rotation and translation of edges from one frame to the matched corresponding edge in other frames using a set of edge descriptors (six parameters of a straight edge in 3D) to update subsequent camera positions. Finally, we compare the result with measurements made from an independent source, the Personal Space Tracker (PST-110) manufactured by Personal Space Technologies.**

***Index Terms*— kinect, personal spacetracker, edge descriptor, 3D edge.**

I. Introduction

We hold that the straight edges abound in man-made environments, such as urban structures, and that this favors the use of strong contrast straight edge features due to their easy identification in images, their insensitivity to illumination changes, noise or textures in the scene. Also, their invariance to viewing direction, and greater immunity to occlusion result from the inherent redundancy arising out of mapping a denser description of the observed static environment than point features can. Paper [1] used edgelets, which are fixed length locally straight portions of an edge as landmark features. These edgelets are tracked in a region of a predetermined width of 15 pixels surrounding the edgelet. Using Hough transforms and least squares line fitting, the edgelet present in the search area closest to the edgelet being tracked is found. [2] have described how straight lines can be added to monocular EKF (Extended Kalman Filter) [3] SLAM in a manner which is both fast and integrated easily with point feature. They detect line features using a fast straight line detector. Another paper [4] used infinite line features to avoid issues arising due to uncertain determination of end points. They detect straight lines by dividing the edge map into 8×8 pixel cells. To track the visible motion between two images [5] used the intersection points between pairs of almost perpendicular finite line segments, which are more stable than end points or the position of line segments along their direction, utilizing a cascade filtering approach that involves applying complex operations only at locations passing an initial test. We have extended this scheme by localizing edges in 3D space allowing a more robust feature extraction.

SLAM is a dynamic problem in which estimation of robot pose and building a map of an unknown environment are done simultaneously. Simple color cameras as in paper [6]-[8] gives RGB images which provide 2D visual information without depth, hence visual SLAM has been worked on in the past as reviewed above. The use of edges as features is not renowned in SLAM research but with the Kinect, the use of 3D edge features becomes both important and feasible. Microsoft Kinect (paper [9] and [10]) is a peripheral device which gives depth estimates through the IR sensor available on it. The device consists of a multi-array microphone, a RGB camera, an infrared projector and an IR sensor. The operating depth range of Kinect is 0.5 meters to 4 meters. The color image which we get from Kinect has 8×3 bits/pixel while the depth image uses 11 bits/pixel.

II. 3D Straight Edge Detection and Extraction

Since the Canny edge detector is susceptible to noise, the raw image provided by the Kinect is first convolved with a first order Gaussian filter to suppress high frequency noise that gives rise to spurious artifacts from the Canny detector. The result of filtering is a slightly blurred version of the original image. From this detector output, we then extract straight lines by putting a limit on their curvature. Then, we dilate these straight line edges so as to enhance their tolerance against noise. A line joining two edgels (straight edge pixels) fully contained in the dilated edge is accepted as a straight line, but even if a single pixel of the line is not in the dilated image, it is rejected.

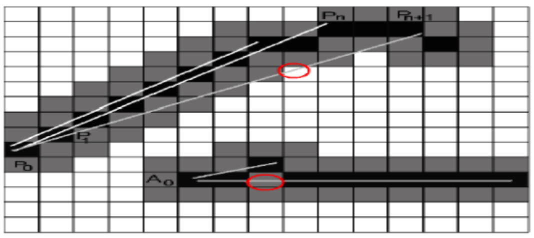


Figure 1. Straight line extraction

Manuscript received February 7, 2013; revised April 24, 2013.

In Fig. 1, the pixel with the red circle is not contained in the dilated image, hence the edge containing it is rejected. We ignore point p_{n+1} and limit the line upto point p_n from point p_0. After having the individual line segments, we look for connected line segments since they are easier to track or locate. Now once we have the connected line segments, we process these lines further only if two distant points on a particular line satisfies equation of 3D straight line.

III. Descriptor for Straight Space Edges

A relationship between the actual depth value and the value obtained from depth map generated by Kinect (normalized to 8-bit values) is established which is given as follows:

$$Z_{K-depth} = 0.1236 \times \tan\left(0.0028255 \times Z_{depth-map} + 1.1863\right) - 8.5 \quad (1)$$

Here $Z_{K-depth}$ is the calculated Kinect depth in meters and $Z_{depth-map}$ is the depth value supplied from Kinect. Fig. 2 shows the comparison between actual and the Kinect depth. The depth information provided by Kinect is used to get the parameters of 3D line. We exploit the concept of projection to determine the edge descriptors. Fig. 3 shows 3D coordinate system with XY as a two dimensional image plane and Z as the third dimension providing depth information. A 3D line is shown having points A and B. The projection of points A and B on XY plane is A_{xy} and B_{xy} by considering zero depth value. Similarly, projecting the points A and B on XZ plane we get projection $A_{xz}B_{xz}$ and considering same X values we get projection $A_{yz}B_{yz}$ in YZ plane respectively.

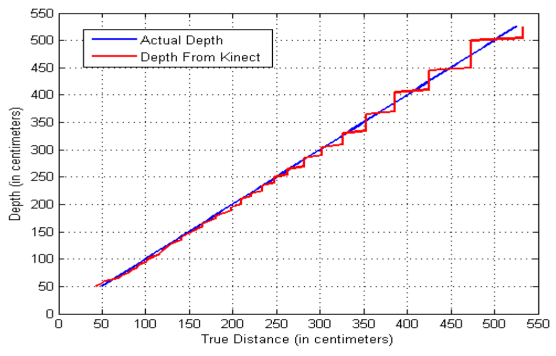


Figure 2. Comparison between Kinect and actual depth.

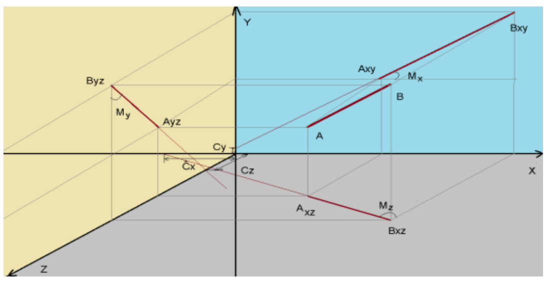


Figure 3. A straight space edge.

All the lines obtained in the above estimation procedure are in 2D image plane and hence can be represented by the equation of a 2D line as:

$$y = m_x x + c_y \quad (2)$$

$$z = m_y y + c_z \quad (3)$$

$$x = m_z z + c_x \quad (4)$$

where m_x, m_y, m_z and c_x, c_y, c_z are the inclinations and interceptions on X, Y and Z axis respectively. These six parameters form the basis of our proceeding calculations.

IV. Edge Tracking Algorithm

We have used template matching algorithm with sliding window technique to establish edge correspondences in two frames. Since matching is performed on the basis of color intensity values and the texture information present in the neighborhood of an edge, we make edge templates through the color images. As shown in Fig. 4, we have created two edge templates, one is left side template and the other is right side template so as to deal with the unpredictable determination arising out of occlusion. The right side of the template of an edge contains the color intensity values of the original image and three parallel right side segments and vice versa for left side template. Now, a one to one mapping which need not involve every single feature present in the edge establishes feature correspondence to match a set of features between two images.

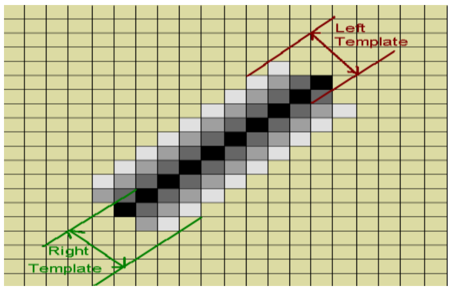


Figure 4. Creating right and left side templates

It compares a template against overlapped target image regions. Sliding through the image, it compares the overlapped patches of size against the template and find sum of squared differences. If T_{1l} is the template of left side edge in image1 and T_{2l} is the same of image2, then the sum of squared error E_l is given by:

$$E_l(x,y) = \sum_{x',y'} \left(T_{1l}\left(x',y'\right) - T_{2l}(x+x',y+y')\right)^2 \quad (5)$$

We set a threshold limit on the error and after thresholding, the minimum of the error provides the best matching.

V. Rotations and Translations

Once edge correspondences are established, we find the camera rotations and translations of a 3D edge in image frame1 so that it can be transformed into its matched corresponding 3D edge in frame 2.

In Fig. 5 shown below the line AB shows a detected 3D line in one image frame and the line CD is the corresponding matched line in next image frame. The projections of line AB and CD on XY plane are a_1b_1 and c_1d_1. Similarly, their projections on YZ plane are a_2b_2 and c_2d_2 and on XZ plane are a_3b_3 and c_3d_3 respectively. Consider the projection of the line on XY plane where m_1 and m_2 are the inclinations of the line a_1b_1 and its corresponding matched edge c_1d_1 to the X axis. C_1 and C_2 are the intercepts of two respective lines on Y axis. We follow three steps to transform a line a_1b_1 to line c_1d_1 (see Fig. 6). Shift the line a_1b_1 to the origin by using translation of amount '$-C_1$'. Then rotate the line a_1b_1 by the amount $m_x = m_2 - m_1$. Finally, translate the line back by amount 'C_2'.The process transforms the line a_1b_1 to c_1d_1. The net effective translation is $C_y = C_2 - C_1$. Similarly, we can obtain m_y, m_z, C_x and C_z where m_y and m_z are the two required rotations of the projection lines a_2b_2 and a_3b_3 into c_2d_2 and c_2d_3 while C_z and C_x are the net effective translations of the two respective lines. Now we set a threshold limit on the intercepts and inclination values such that for n^{th} image frame, we calculate its rotations and translations to $(n+1)^{th}$, $(n+2)^{th}$…$(n+m)^{th}$ image frames till we get a considerable rotation and translation between two image frames to satisfy the threshold limit. Finally, a voting is carried out to plot histograms for taking the majority decision among the six selected parameters individually, since there exist a few mismatched edge pairs and also because of rounding of errors in the calculations.

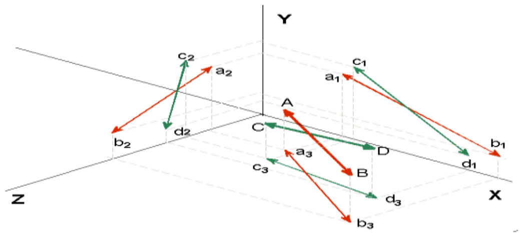


Figure 5. A straight space edge.

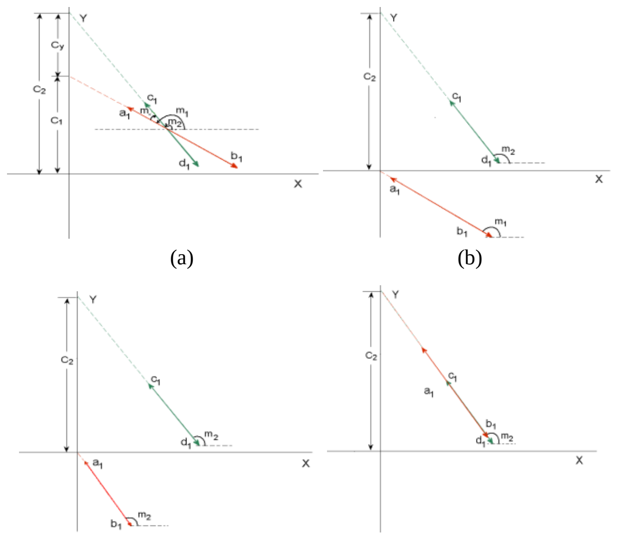


Figure 6. (a) Projection of lines in YZ plane; (b) Shift the line a_1b_1 to the origin; (c) Rotate the line a_1b_1 by the amount $m_x = m_2 - m_1$; (d) Translate back the line a_1b_1 by amount 'C_2'.

VI. Estimation of New Camera Position

In this last step of computation, we find the new camera position and orientation based on its previous position and update the map. We have used the Euler transformation for this purpose. Let's say our present camera position is (x_0, y_0, z_0) and the three planer camera rotations and translations are m_x, m_y, m_z and c_x, c_y, c_z respectively. Then R_x, R_y, R_z are the rotation matrices with axes of rotation being X, Y, Z respectively. The overall rotation matrix R and the translational matrix T are given as:

$$R = R_x \times R_y \times R_z \tag{6}$$

$$T = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} \tag{7}$$

Now the new camera position (x_1, y_1, z_1) is calculated as

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = R \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + T \tag{8}$$

The new camera position is now updated to build the map.

VII. Implementation and Result

Implementation of SLAM is done using C and open CV library for computer vision on AMD Athlon (tm) 64X2 dual core processor machine with 2.4 GHz clock rate and main memory of size 2 GB. Kinect is used with open Kinect platform in ubuntu 11.04 which gives color image frame of size 640 × 480 and depth image frame of 11 bits and of same size. In Fig. 7 we have first shown two images of a test video which are 10 frames apart and are used to find edge correspondences. For visualizing locali

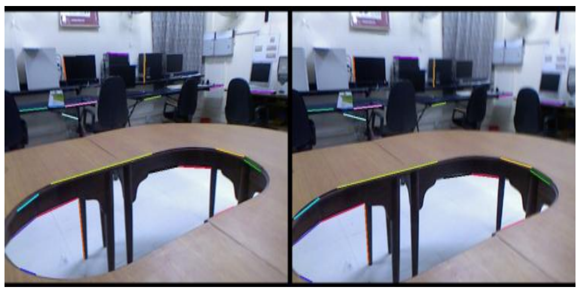


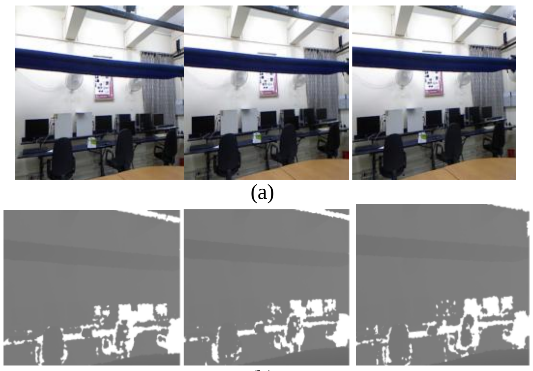
Figure 7. Shows indoor scenario of lab where the colorful lines shown are the detected lines in 3D.The matched lines in the two frames are shown by same color.

zation output, we have selected a dataset (Fig. 8) which have been taken in the lab with a two dimensional camera motion. The result is then compared to ground truth trajectory generated by PST with time. The blue line shows the result of our algorithm and the red line is the ground truth trajectory obtained through PST (see Fig. 9).

Now to analyze the behavior of the proposed algorithm, we have calculated system transfer function in relation to the ground truth generated by personal space tracker which uses infrared rays to detect its target and supplies the actual 3D position (X, Y, Z coordinates) of an object with millimeter accuracy. We have fixed the location of PST such that the IR reflective stickers mounted on the back of Kinect are visible to it for all camera motions. The actual ground truth position of PST is given by x_p and the result of our algorithm is denoted by y.

If we denote system transfer function as h then $y = x_p * h$. Here '$*$' is convolution operator. Taking DFT we get system transfer function as: $H(\Omega) = Y(\Omega)/X(\Omega)$.

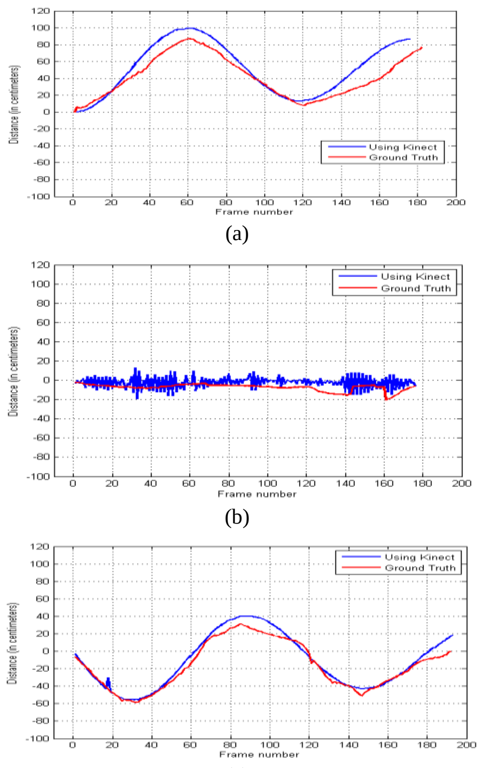
We have plotted the magnitude and phase response of the transfer function individually for X, Y and Z trajectories (see Fig. 10).



(a)

(b)

Figure 8. Camera has been moved in X and Z directions keeping Y coordinate constant (a) Data set containing color images for slam computation;(b) Data set containing depth images for slam computation.

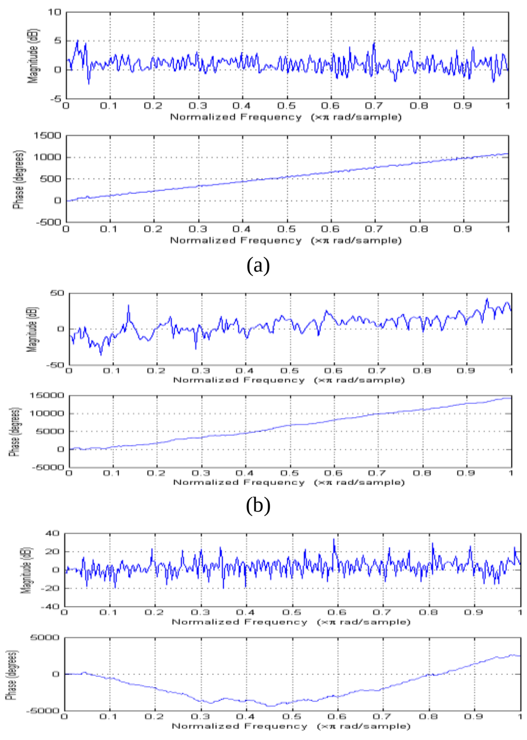


(a)

(b)

(c)

Figure 9. Plots for SLAM computation with Y axis as camera position in one specific direction and X axis as time. (a) Plots of camera trajectories in X direction; (b)Plot for camera trajectories in Y direction; (c)Plot for camera trajectories in Z direction



(a)

(b)

(c)

Figure 10. Magnitude and phase plot of the system transfer function $H(\Omega)$ for data-set (a) Calculated frequency response for X coordinates of data set; (b)Calculated frequency response for Y coordinates of data set ;(c) Calculated frequency response for Z coordinates of data set.

VIII. Conclusion

We have presented our preliminary results to track straight edges in 3D space. To find the new camera position, we calculate the rotations and translations of straight space edges from one image frame to another using simple geometry. Parameters of a 3D line (intercepts and inclinations) are used as a descriptor, which makes the whole system computationally efficient. Kinect is used for the purpose of providing 2D color video as well as depth information. As progress is apparent, the depth-map obtained from Kinect can be used directly for 3D line detection by applying a suitable smoothing technique on raw depth images. With the addition of statistical approach to the system it can predict new camera states based on previous states such as 'Kalman filtering'. The proposed algorithm can be further used for 3D reconstruction of the scene and a 'Walk through' can be generated.

References

[1] E. Eade and T. Drummond, "Edge landmarks in monocular slam," *Image Vision Comput*, vol. 27, pp. 588-596, Apr 2009.

[2] P. Smith, I. D. Reid, and A. J. Davison, "Real-time monocular slam with straight lines," in *BMVC*, 2006, pp. 17-26.

[3] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1052-1067, June 2007.

[4] M. Chandraker, J. Lim, and D. Kriegman, "Moving in stereo: Efficient structure and motion using lines," in *Proc. IEEE 12th International Conference on Computer Vision*, Oct 2009, pp. 1741-1748.

[5] S. Jain, K. S. Venkatesh, and A. Mukerjee, "Edge-feature tracking," M. Tech Thesis, October 2011.

[6] P. Jensfelt, D. Kragic, J. Folkesson, and M. Bjorkman, "A framework for vision based bearing only 3d slam," in *Proc. IEEE International Conference on Robotics and Automation*, May 2006, pp. 1944 -1950.

[7] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Monocular vision based slam for mobile robots," in *Proc. 18th International Conference on Pattern Recognition*, 2006, vol. 3, pp. 1027-1031.

[8] L. Goncalves, E. di Bernardo, D. Benson, *et al.*, "A visual frontend for simultaneous localization and mapping," in *Proc. IEEE International Conference on Robotics and Automation*, April 2005, pp. 44 - 49.

[9] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3d visual slam with a hand-held camera," *in Proc. RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, April 2011.

[10] F. Endres, J. Hess, N. Engelhard, J. Sturm, and W. Burgard, "Online-6d-slam für rgb-d-sensoren," *Automatisierungstechnik*, 2012.

Vaishali Ailani is presently working for Multimedia Subsystem team in QUALCOMM Technologies Inc India. She received her Master's degree in Signal Processing, Communication and Networking from Indian Institute of Technology Kanpur, India in 2012 and her B.Tech. in Electronics and Communication engineering from National Institute of Technology, Raipur, India in 2010. Her research interests include Image Processing, Computer Vision and Wireless communication.

Disha Prakash received her B.Tech Degree in Electronics and Instrumentation from Maharana Institute of Professional Studies, Kanpur. Currently, she is working as a project associate in the department of Electrical Engineering and A.C.E.S, Indian Institute of Technology, Kanpur. Her research interests include Image Processing and Computer Vision.

K. S. Venkatesh received his B.E degree in Electronics from Bangalore University, M.Tech degree in Communication and PhD in Signal Processing from Indian Institute of Technology, Kanpur. Currently, he is working as a professor at the Electrical engineering department in Indian Institute of Technology, Kanpur. His research interests include Image Processing, Video Processing, Computer Vision and Vision applications in Navigation and Robotics.