Localization for Mobile Robot Using Vertical Lines

Chang-Hun Kang, Hyunsik Ahn

Department of Robot System Engineering, Tongmyong University of Information Technology
(Tel : +82-51-610-8356; E-mail: hsahn@tmic.tit.ac.kr)

Abstract: In this paper, we present a self-localization method for mobile robots using vertical line features of indoor environment. When a 2D map including feature points and color information is given, a mobile robot moves to the destination, and acquires images by one camera from the surroundings having vertical line edges. From the image, vertical line edges are detected, and pattern vectors meaning averaged color values of the left and right region of each line segment are computed. The pattern vectors are matched with the feature points of the map using the color information and the geometrical relationship of the points. From the perspective transformation of the corresponded points, nonlinear equations are derived. Localization is carried out from solving the equations by using Newton's method. Experimental results show that the proposed method using mono view is simple and applicable to indoor environment.

Keywords: mobile robot, localization, matching, line edge, correspondence, 3D vision

1. INTRODUCTION

Recently, there is much concerning on mobile robot working with human at home. Basically, mobile robots need to recognize surroundings, find a path, and localize his position[1]. The topic of self-localization is can be divided into two sub-topics, position tracking, which assumes the initial location and follows current position, and global localization, which is being able to localize without any information of the previous pose[2-5]. In the global localization, vision based methods, using stereo vision, omni-directional view, and mono view, have been intensively studied. Stereo vision has advantages detecting depth from the surroundings directly, but they demand complicated hardware and much processing time[6]. Omni-directional vision method using a conic mirror can be an alternative solution, but it has problems of distorted view and low resolution[7]. Mono view methods using artificial landmarks are able to estimate the current position by recognizing the pattern of landmarks, but we have to attach them for detection[8].

Generally, indoor environments are consists of horizontal and vertical line features such as doors, furniture, and so on. In this paper, we propose a self-localization method just using line features of circumstance when a 2D global map is provided. From the image acquired from the camera attached on the robot, we detect vertical lines and their pattern vectors, match the vectors with the feature points of the map. Then the current pose of the robot (a,b,θ) is estimated by solving nonlinear equations derived from perspective transformation.

In the next of this paper, we will describe a line feature detection method and a matching algorithm, and detail the proposed self-localization method. The next parts will show the experimental results and conclude the paper.

2. LOCALIZATION USING VERTICAL LINES

Fig. 1 shows the localization procedure. First, we design a 2D map consisting of the position of corners and the colors among them. On the map, a user can indicate destination, find a path covering from the present position to the goal. When the robot moves to the destination, a camera attached on the robot acquires images and detect line segments by using the proposed method. If the number of features is larger then 3, we match the features of the image to the points of the map, and compute the location of robot (X,Y,θ) by proposed localization method. Then, we compute the distance between previous and current positions. If it is larger then a threshold, because we estimate it is uncertain, flows are to go to the initial step.

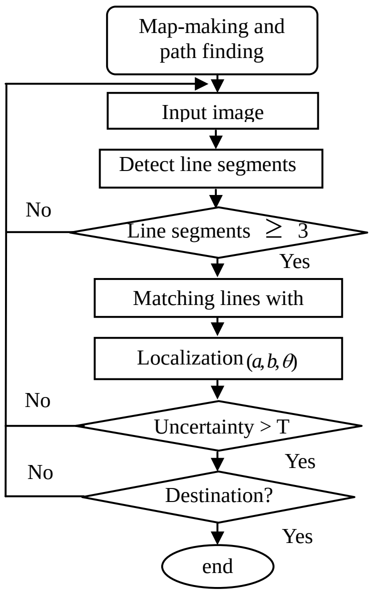

Fig. 1. The flowchart of self-localization

2.1 Line feature detection

For detecting the location of robot, users first detect vertical line features from the image. The robot has a camera at the front of the robot and acquires images while it moves. The algorithm for detecting vertical lines is like as follow.

- We vertically apply Sobel edge operator to the image, and make a binary image applying a threshold value. Then we project the edge image to the horizontal axis to form an accumulated histogram as Fig. 2.
- Projected histogram is averaged by an one dimensional mask and divided with a threshold value to suppress noises. And we find local maxima from the histogram and index them as feature points.

- By comparing the vertical lines defined by the feature points with the intensity of the edge image, vertical lines are determined.
- Since the camera is horizontally attached, the vanishing point is vertically located at the middle of image. If the lowest vertex of a line feature is higher then the central position, we conclude that it is not contacted with the floor and delete it from the candidates.
- The points crossing vertical lines and U axis of image plane indicate the feature points $(x_1, x_2, \cdots, x_n)$. They are compared with discontinuous features such as corners and door's edges of the map, correspondence is estimated.

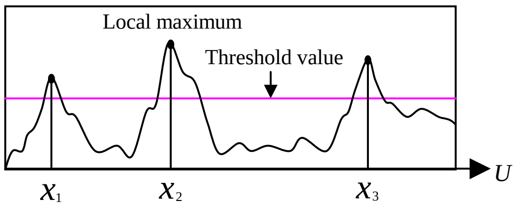

Fig. 2. Projected histogram and a local maximum

2.2 Correspondence of features

We need to match the feature points with the map. It means that the feature points $(x_1, x_2, \cdots, x_n)$ should be corresponded with those of map $(g_1, g_2, \cdots, g_n)$. In this paper, we consider not only the positions but also the color values of each side of the line features. A feature vector is defined by the color values of the right and left regions of the line, and used them for determining correspondence. The geometrical information of the line features in the map also contributes to improve accuracy of localization.

The right and left regions should be defined for computing the feature vectors. In this paper, we determine them using region growing. In the case of the right region, a centered pixel is defined as the pixel located at the middle of vertical line with the V axis, and the right of the vertical edge with the U axis. 8-neighbors $N6$ of P are compared to P itself, if the differences are smaller than a predefined threshold value, the neighbor pixels are classified as one of the right region. This procedure is applied to all of the neighbor pixels, we can determine the right region. With the same method is applied to the left side, the left region is also defined. A floor region is also detected by region growing with the seed pixel located at the bottom of the image.

From the acquired both regions, feature vectors are defined. There can be various features extracted from the line features and their neighboring regions. We take the mean of the color values of each region as a feature vector. For reducing the influence of the illumination of light source, RGB(Red, Green, Blue) values are transformed to the HSI(Hue, Saturation, Intensity) model, and hue and saturation values of the pixel are used as two components of the vector. Hue(H) and saturation(S) can be obtained by Eqs. (1) and (2).

$$H = \cos^{-1}\left[\frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right] \cdot \qquad (1)$$

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right] \cdot \qquad (2)$$

Feature vectors of right and left regions, R_i and L_i, are defined as Eqs. (3) and (4). Where r_1 and r_2 are hue and saturation values of the right region, and l_1 and l_2 are ones of the left respectively.

$$L_i = \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \qquad i = 1,2,\cdots,n \qquad (3)$$

$$R_i = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \qquad i = 1,2,\cdots,n \qquad (4)$$

In this paper, the feature points of the map and the image are matched with feature vectors and the geometrical relationships among the points. The matching procedure can be described as follow with Fig. 3.

- First we check whether line features meet the floor region, if they do, we classify them as contacted lines such as x_1, x_2, and x_3, otherwise we identify them as non-contacted lines.
- In the case of contacted line, we divide the right and left regions of the line into visible or occluded regions according to the pattern of the lowest vertex. Horizontal Sobel operator is applied to the right and left sides of the vertex. If there exists a kind of edge, we defined the region as a visible region such as circled l_1 to r_4 in Fig. 3, but an occluded region.
- The vectors of both regions are matched with the feature points of the map. In matching procedure, we use a priority. First, the lines that are visible in right and left regions are checked. Secondly, the lines having one visible region are considered. Then non-contacted lines are examined.
- The lines of image are compared with the feature points of the map by utilizing their positions and the color values of the region, and the candidates of the matching pairs are determined. The number of pairs is larger than 2, the correspondence of neighbor lines are investigated with geometrical relationship. If a point of map is a part of convex object like x_1 and x_4 of the Fig. 3, we have to consider that both regions should be inverted.

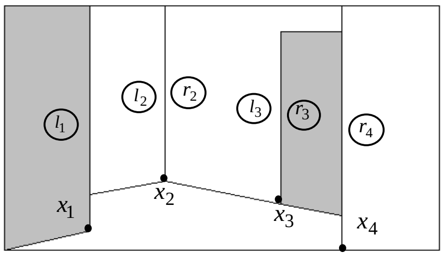

Fig. 3. Floor contacted lines and visible regions

2.3 Localization algorithm

For Localization, the coordinates of feature points $(x_1, x_2, \cdots, x_n)$ of the image are corresponded with the coordinates (Cx_1, Cy_1), $(Cx_2, Cy_2), \cdots$, (Cx_n, Cy_n) of the map. With the *n* points of each coordinate, we induce *n* number of perspective equations. The camera coordinates and the world coordinates are related with the rigid transformation of translation and rotation. The equations of perspective and rigid transformations of the coordinates come out a system of nonlinear equations. Newton's method is used for computing the location of robot (a,b,θ) from the equations. Fig. 4 shows the perspective transformation of camera coordinates. Fig. 5 shows the relationship between global and camera coordinates. Where, the definitions of characters are as follow.

(X,Y,Z)	: Global coordinates
$(Gx_i, Gy_i)\quad i=1,2,\cdots,n$	: Feature points in the global coordinate system
(Xc,Yc,Zc)	: Camera coordinates
$(Cx_i, Cy_i)\quad i=1,2,\cdots,n$	: Feature points in the camera coordinate system
$(x_1, x_2, \cdots x_n)$	: Feature points of image plane
λ	: Focal length of camera
(a,b,θ)	: Translation and rotation of robot
(U,V)	: Coordinates of image plane
n	: The number of features

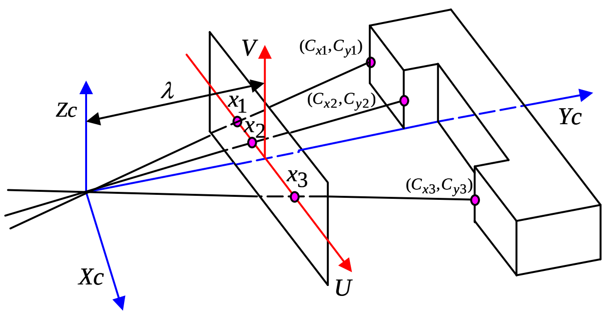

Fig. 4. Perspective transformation of camera coordinates

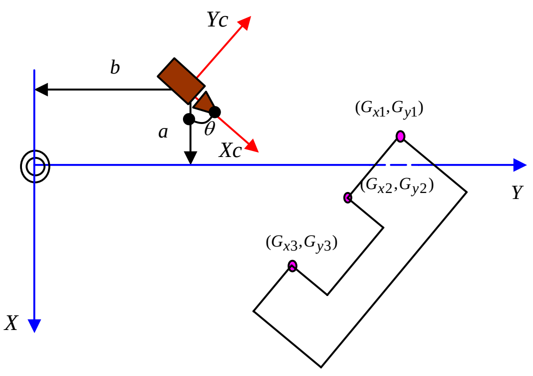

Fig. 5. Global and camera coordinates

Camera coordinates can be transformed to world coordinates by a rigid transformation T. Where the moving of robot is restricted with the translation with X and Y axes and rotation about the Z axis, accordingly, the pose of the robot can be defined by (X,Y,θ), and G_{zi}, C_{zi} are zero. From rigid transformation,

$$\begin{bmatrix} Gx_i \\ Gy_i \\ 0 \\ 1 \end{bmatrix} = T \begin{bmatrix} Cx_i \\ Cy_i \\ 0 \\ 1 \end{bmatrix} \quad i=1,2,\cdots,n. \tag{5}$$

The transformation T, as shown in Fig. 5, can be defined as

$$T = [T_{trans}][T_{z\theta}] = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

Eq. (5) can be rewrite as

$$\begin{bmatrix} Cx_i \\ Cy_i \\ 0 \\ 1 \end{bmatrix} = T^{-1} \begin{bmatrix} Gx_i \\ Gy_i \\ 0 \\ 1 \end{bmatrix} \quad i=1,2,\cdots,n. \tag{7}$$

Where (G_{xi}, G_{yi}) can be defined from the matching described in the previous section. For localization, the problem is just computing the parameters (a,b,θ). We can derive the perspective transformation, as shown in the Fig. 4,

$$Cx_i = \frac{x_i}{\lambda} Cy_i \quad i=1,2,\cdots,n. \tag{8}$$

We can induce nonlinear equations with variables (a,b,θ) from Eqs. (7) and (8). If the number of feature points is more than 3, we can get the solution of the equations by using Newton's method. Vector function *F* has *n* functions according to the matched feature points between image plane and the global coordinates.

$$F(a,b,\theta) = (f_1(a,b,\theta), f_2(a,b,\theta), \cdots, f_n(a,b,\theta))^t. \tag{9}$$

We represent variables (a,b,θ) as a vector P, the system of nonlinear equations can be expressed by $F(a,b,\theta)=0$. Jacobian matrix $J(a,b,\theta)$ is

$$J(a,b,\theta) = \begin{bmatrix} \frac{\partial f_1(a,b,\theta)}{\partial a} & \frac{\partial f_1(a,b,\theta)}{\partial b} & \frac{\partial f_1(a,b,\theta)}{\partial \theta} \\ \frac{\partial f_2(a,b,\theta)}{\partial a} & \frac{\partial f_2(a,b,\theta)}{\partial b} & \frac{\partial f_2(a,b,\theta)}{\partial \theta} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n(a,b,\theta)}{\partial a} & \frac{\partial f_n(a,b,\theta)}{\partial b} & \frac{\partial f_n(a,b,\theta)}{\partial \theta} \end{bmatrix}. \tag{10}$$

Newton's method to find the solution of the nonlinear equations $F(a,b,\theta)=0$ is as Eq. (11), when initial value $P^{(0)}$ is given.

$$p^{(k)} = p^{(k-1)} - \left[JP^{(k-1)}\right]^{-1} F(P^{k-1}), k \ge 1 \tag{11}$$

To obtain solution, Eq. (11) is computed recursively. If variation of $P^{(k)}$ is in a limit value, we can end the computation and get $P^{(k)}$ as the solution. The computed solution $P^{(k)}$ means the location and rotation of the robot (X,Y,θ).

3. EXPERIMENTAL RESULTS

The proposed self-localization method was applied to a real indoor environment, a laboratory. Fig. 6 shows the mobile robot, TIROB-II, and a camera attached to the front of that. Fig. 7 shows the procedures of detecting vertical line features; Fig. 7(a) is captured image from the camera, Fig. 7(b) shows the result of vertical edge detection, Fig. 7(c) shows the projected histogram of the edge image, and Fig. 7(d) shows the detected line segments. From the matching algorithm, pattern vectors were computed and matched with feature points of the map. The current pose of the robot (a,b,θ) is computed by solving nonlinear equations derived from the perspective transformation of the matched feature points. Table 1 shows the results of real and estimated positions of the robot. From the table, we were able to find the doubled standard deviation of errors between the real and the computed coordinates were less then 33mm and 54mm in translation of X and Y axes respectively, and 0.61 ° in rotation. We can conclude it is enough to apply the proposed method to mobile robots moving in indoor environment. Fig. 8 shows the amount of errors when the robot approaches though Y axis to the objects, furniture and a wall, which have vertical line features. When the robot approaches to the object, the errors are undulated. It means if the camera is near the features, the errors are diminished generally. However, in the case of real objects including dull edges, it is possible the vague borders make some errors. Fig. 9 shows the moving of the robot (square) and the estimated location (circle) in the map, when it approaches the destination.

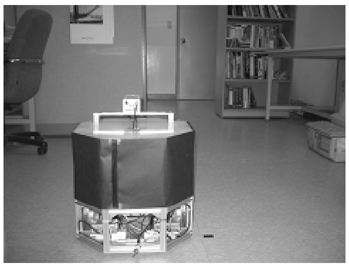
Fig. 6. Mobile robot and test environment

4. CONCLUSIONS

In this paper, a self-localization method using vertical line features with mono view was proposed. From the image acquired from a camera attach on the robot, line features were obtained by finding local maxima form the vertically projected histogram after vertical edge operation. Then the right and left pattern vectors are defined for identify lines, and matched with the feature points of the map. After estimating correspondence within the points, we induced a system of nonlinear equations with perspective and rigid transformation of the matched points. Newton's method was used to solve the equations for obtaining the position (X,Y,θ) of the robot. The experimental results showed that the error between the real and estimated position is acceptable to home robot. We can conclude that the proposed method using mono view is simple and applicable to self-localization of mobile robots working at indoor environment.

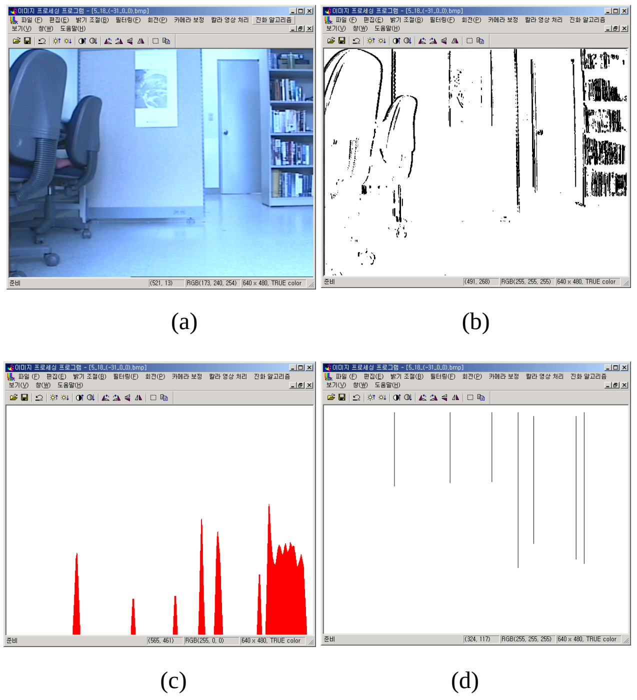

Fig. 7. The procedures of detecting vertical line features, (a) original image, (b) vertical edge, (c) projected histogram, and (d) detection of vertical line features

REFERENCES

[1] R. Cipolla and N. J. Hollinghurst, "Human-robot Interface by Pointing with Uncalibrated Stereo Vision," *Image and Vision Computing*, vol. 14, no. 3, pp.171-178, 1996.

[2] D. Lowc and J. Little, "Mobile Robot Localization and Mapping with Uncertainty Using Scale-Invariant Visual Landmarks," *The International Journal of Robotics Research*, vol. 21, no. 8, pp. 735-758, Aug., 2002.

[3] A. J. Davision and D. W. Murray, "Simultaneous Localization and Map-Building Using Active Vision," *IEEE Transactions on Patten Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865-879, Jul., 2002.

[4] P. Ranganathan, J. B. Hayet, M. Devy, S. Hutchinson and F. Lerasle, "Topological Navigation and Qualitative Localization for Indoor Environment Using Multi-sensory Perception," *Robotics and Autonomous*, vol. 41, pp. 137-144, 2001.

[5] J. Fredslund, M. J. Mataric, "Robot Formations Using Only Local Sensing and Control," *Proceedings, International Symposium on Computational Intelligence in Robotics and Automation*, Canada, Jul., 2001.

[6] In-Hyuk Moon, "Automatic Extraction of Stable Visual Landmarks for a Mobile Robot under Uncertainty," *Journal of Control, Automation and Systems Engineering*, Vol. 7, No 9, pp. 759-765, Sep., 2001.

[7] A. Rizzi and R. Cassinis, "A Robot Self-localization System Based on Omni-directional Color Images," *Robotics and Autonomous System*, vol. 34, pp. 23-38, Apr., 2001.

[8] E. Stella and A. Distante, "Self-location of a Mobile Robot by Estimation of Camera Parameters," *Robotics and Autonomous System*, vol. 15, pp. 179-187, 1995.

Table 1. Real positions and errors

Real position (mm)				Measured position (mm)		
Number	X	Y	Angle	X	Y	Angle
1	0	0	0	23.7919	46.1322	0.0416
2	-160	100	0	32.5107	41.5437	1.5305
3	-160	200	0	49.9005	58.2881	1.2496
4	-160	400	0	34.5437	74.8281	1.3929
5	-160	600	0	37.6711	61.4102	1.2038
6	-160	800	0	29.3517	43.8672	1.3178
7	-160	1000	0	26.5774	100.371	1.3757
8	-160	1200	0	30.4663	18.4712	1.3843
9	-160	1400	0	18.5924	96.3906	1.4933
10	-160	1600	0	14.1872	93.7453	1.3184
11	-160	1800	0	9.3829	9.4685	2.004
12	0	660	0	20.2111	7.8468	3.324
13	0	1150	0	34.6217	72.9892	2.292
14	0	1555	0	24.3689	44.7894	1.834
15	0	2005	0	15.6641	35.6003	0.263
16	-650	380	0	50.0889	88.5548	0.619
17	-630	660	0	15.8471	32.2951	0.894
18	-560	1045	-48	4.1574	27.5667	1.78
19	-465	1300	-45	36.0135	59.975	0.84
20	-375	1465	-47	28.0305	41.1798	2.82
21	-285	1740	-37	11.2763	25.8148	0.83
22	-190	2125	-32	7.3323	73.2939	1.69
23	-75	2435	-22	18.2157	70.0582	1.15
24	-25	2715	-10	19.2574	44.0739	3.24
25	165	3034	-23	10.0399	70.4309	1.97
26	325	3455	-27	80.639	66.9437	1.45
27	370	3915	-15	9.0458	11.0485	2.5
2σ of errors				32.8329	53.8091	1.58986

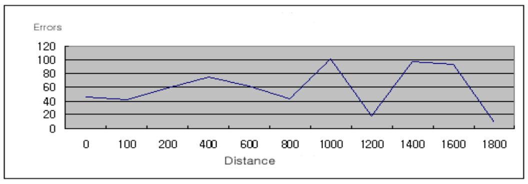

Fig. 8. Errors through Y axis

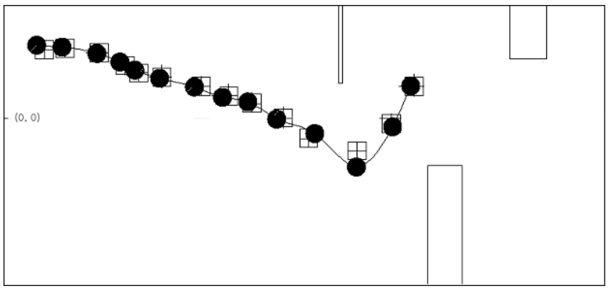

Fig. 9. The moving of robot and the result of localization in the given map