コンピュータビジョン 76－2
(1992. 1. 23)

Vision-based Behaviors for Indoor Mobile Robots

In So Kweon, Kazunori Onoguchi, Mutsumi Watanabe, Yoshinori Kuno
Research & Development Center
Toshiba Corporation
1 Komukai, Toshiba-Cho, Saiwai-Ku
Kawasaki, 210, Japan

December 22, 1991

Abstract

In this paper, we present vision-based behaviors, such as wall-follower *and* intersection-tracker, *for an indoor mobile robot.*

We develop a method to extract vanishing points and two associated edges which are the projection of straight lines lying on the floor. This information effectively describes the geometrical relationship between camera coordinates and 3-D scene coordinates. Wall-following behavior uses this geometrical relationship to guide the robot along the wall. Intersection-tracking behavior extracts and tracks two vertical lines using a vanishing point and edge segments.

We present experimental results in a real office environment using a mobile robot which was developed by Toshiba.

1 Introduction

Conventional mobile robot navigation systems attempt to build two or three dimensional world models using multiple sensors and then plan routes for the robot using this model [12]. A serious problem in this approach is reliability. To improve the reliability of system, many researchers have integrated information from different sensors to build a more accurate (or complete) model for safe navigation [5, 9]. As a result, systems cannot work very well in complex environments.

On the other hand, a behavior-based approach is to decompose the system into individual modules, each of which is responsible for one behavior to be performed by the entire system [3, 6, 10]. Each behavior contains a complete path, from sensing to action, and is executed in a completely parallel manner. A behavior-based approach is more reliable than the conventional approach. Since even if one module fails, other behaviors can still produce meaningful actions for the robot.

Most behavior-based systems have used simple ultrasonic, tactile or proximity sensors [6, 7]. However, vision-based behaviors are seldom found in the literature because of the computational complexity associated with visual data processing [13].

In this paper, we present vision-based behaviors which can contribute to making robots navigate in indoor environments. In a man-made environment, many lines are parallel (e.g., the edges of a wall) and provide very useful information about the scene. To handle the computational complexity, we extract the projection of these lines from input video images and our *wall-following* and *intersection-tracking* behaviors use this information to guide a mobile robot.

In Section 2, we review vanishing points and describe an efficient method for their extraction from input video images. Wall-following and intersection-tracking behaviors are described in Sections 3.1 and 3.2, respectively. In section 4, we present experimental results obtained with a mobile robot which we have built at Toshiba.

2 Vanishing points

In this section, we briefly review vanishing points. A complete treatment of vanishing points can be found in [8].

The perspective projection of a set of parallel 3-D lines onto an image plane is a set of lines meeting at a common point, called a vanishing point [8]. Let's represent a 3-D line by

$$\mathbf{P} = \mathbf{U} + \lambda \mathbf{V} = (U_x + \lambda V_x, U_y + \lambda V_y, U_z + \lambda V_z) \quad (1)$$

where $\mathbf{U} = (U_x, U_y, U_z)$ and $\mathbf{V} = (V_x, V_y, V_z)$ are a position and a direction vector, respectively.

The perspective projection of the point, $\mathbf{P}$, on the image plane gives an image point (see Fig. 1):

$$\mathbf{p} = (f\frac{U_x + \lambda V_x}{U_z + \lambda V_z}, f\frac{U_y + \lambda V_z}{U_z + \lambda V_z}), \quad (2)$$

where f indicates the focal length of the camera.

In the limit as λ fades to infinity, the point $\mathbf{p}$ becomes a vanishing point:

$$\mathbf{pv} = (x_v, y_v) = (f\frac{V_x}{V_z}, f\frac{V_y}{V_z}). \quad (3)$$

From Eq. 3, vanishing points are dependent on direction of line.

Vanishing points are normally detected by projecting image lines or intersection points between lines onto an accumulator space [11, 2]. Accumulator space approaches require a projection process to transform lines on an image plane to curves in an accumulator space (e.g., a great circle in the commonly used Gaussian sphere) as well as a search process for vanishing points in the accumulator space [2]. As a result, accumulator space approaches are not applicable for a real-time mobile robot navigation. In our work, we use the image plane as an accumulator space and directly search for vanishing points without a projection process.

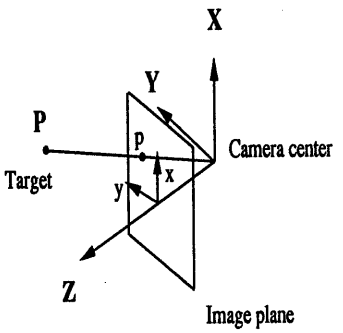
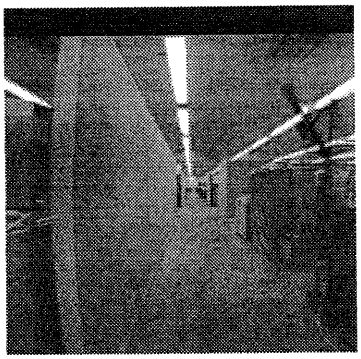
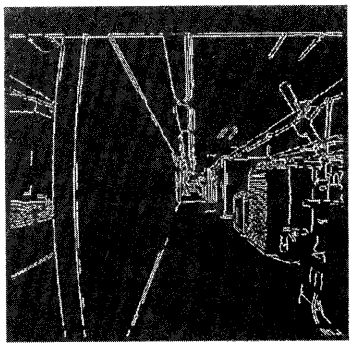


Figure 1: The coordinate system.



(a) An input image



(b) Extracted edges

Figure 2: Indoor scene.

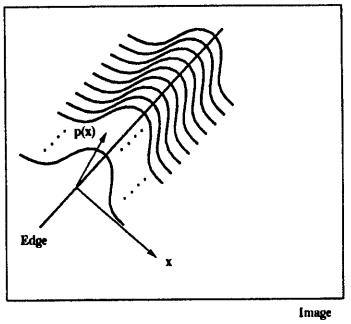


Figure 3: A probability distribution function of vanishing points along edge segments.

We can extract vanishing points by using the fact that a set of parallel lines in the world intersect at a vanishing point in the image. However, the image lines will never meet at one point because of image noise and the errors in localizing lines in the image. We represent the uncertainty in edge location using the 1-D Gaussian distribution along the direction perpendicular to each straight line (computation of σ is described below). Therefore, the search for vanishing points can be easily accomplished by finding a small neighborhood in the image plane intersected by a sufficient number of straight lines. This process is equivalent to dilating edge lines and searching for intersection areas.

The algorithm to extract vanishing points works in the following steps:

1. Extract edges (see Figure 2 for the extracted edges from a real image);
2. Along each edge line segment, compute the belief of vanishing points around each pixel point of edge segment (see Fig. 3) by:

$$p(x) = (2\pi\sigma^2)^{-1/2} exp[-\frac{x^2}{2\sigma^2}]; \quad (4)$$

where σ is inversely proportional to the length of edge segment.

3. Update the belief for pixel points which already have an estimate of the belief from other edge segment:

$$p(x) = p_{new}(x) + p_{old}(x). \quad (5)$$

4. Find the maximum belief point.

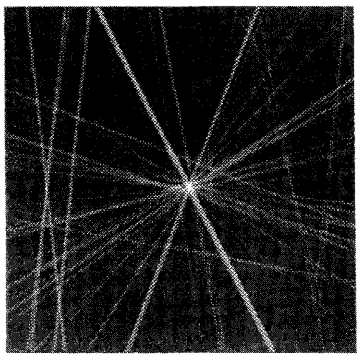
Fig. 4a shows the voting results for a vanishing point.

Fig. 4b shows the experimental results of the extraction of a vanishing point.

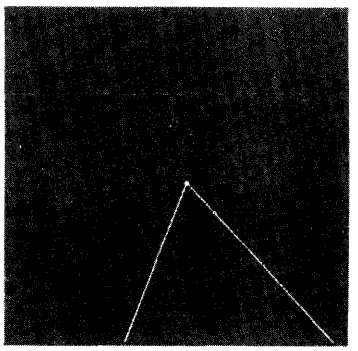
3 Vision-based behaviors

3.1 Wall-following behavior

Part of an indoor environment can usually be described by sets of parallel 3-D lines (e.g., walls and corridors), which



(a) Voting result



(b) Extracted vanishing point

Figure 4: Expeirmental results of vanishing point extraction from a real image.

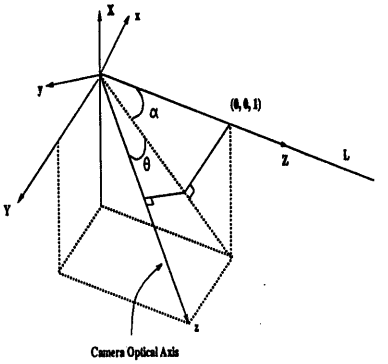


Figure 5: A vanishing point and the robot's steering angle.

form a vanishing point and the associated lines in the image. Since this vanishing point specifies the 3-D orientation of these parallel lines, a mobile robot can be aligned with parallel lines using this vanishing point - wall-following behavior.

We can relate the image coordinates of a vanishing point to the transformation between the camera and world coordinate systems as follows: Consider a 3-D line L and choosc the world coordinate system so that Z-axis is aligned with this line, as shown in Fig. 5. In the world coordinate system , therefore, the unit directional vector of the line is $(0, 0, 1)$. In the camera coordinate system, this becomes:

$$\begin{aligned} \mathbf{V} &= (V_x, V_y, V_z) \\ &= (\sin\alpha, \cos\alpha \sin\theta, \cos\alpha \cos\theta). \end{aligned} \tag{6}$$

where θ and α indicate the pan and tilt angle of the camera with respect to the world coordinate system.

From Eqs. 3 and 6, we obtain a relationship between the camera coordinates and a vanishing point:

$$\mathbf{p_V} = (x_v, y_v) = (f\frac{\tan\alpha}{\cos\theta}, f\tan\theta). \tag{7}$$

The y-coordinate value of a vanishing point, y_v, is equal to zero when the line L is aligned with the camera optical axis (i.e., $\theta = 0$). Therefore, a robot can be aligned with respect to the line L, when the transformation between the camera and the robot coordinate systems is specified. When y_v is not equal to zero, the angle needed to align the robot with the line - steering angle - becomes:

$$\theta = \arctan\frac{y_v}{f}. \tag{8}$$

The x-coordinate value of a vanishing point in the image can now be used to compute the tilt, α, of the camera. From Eq. 7, we can compute the camera tilt angle as:

$$\alpha = \arctan(\frac{x_v\cos\theta}{f}) = \arctan(\frac{x_v\sin\theta}{y_v}) \tag{9}$$

where θ is given by Eq. 8.

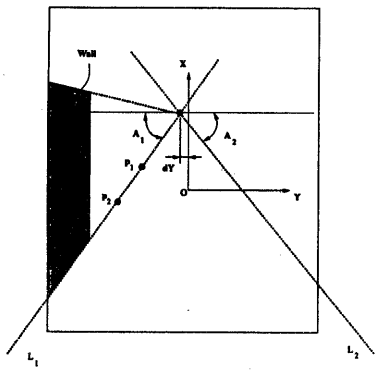


Figure 6: A vanishing point and the robot position.

We will now show how vanishing points are related to the relative position of the robot with respect to the parallel lines in the world.

Let's assume a wall consisting of parallel 3-D lines, as shown in Fig. 6. Without loss of generality and for simplicity, the robot is assumed to align with the wall (i.e., $y_v = 0$ in Eq. 3 and $\mathbf{V} = (0, 0, 1)$) and the line, L_1, passes through a point, $\mathbf{U} = (U_x, U_y, U_z) = (U_x, U_y, 0)$. From Eq. 1 and the above relationships, points on this 3-D line satisfies:

$$\mathbf{P} = (U_x, U_y, \lambda), \tag{10}$$

where λ is the distance along Z-axis.

By perspective projection,

$$\mathbf{p} = (x, y) = (f\frac{U_x}{\lambda}, f\frac{U_y}{\lambda}). \tag{11}$$

As shown in Fig. 6, consider two points, $\mathbf{p_1}$ and $\mathbf{p_2}$, from the line:

$$\begin{aligned} \mathbf{p_1} &= (x_1, y_1) = (f\frac{U_x}{\lambda_1}, f\frac{U_y}{\lambda_1}) \\ \mathbf{p_2} &= (x_2, y_2) = (f\frac{U_x}{\lambda_2}, f\frac{U_y}{\lambda_2}). \end{aligned} \tag{12}$$

From Eq. 12, we can derive:

$$\tan A_1 = \frac{\Delta x}{\Delta y} = \frac{U_x}{U_y} \tag{13}$$

where $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$.

In a similar way, we can also compute $\tan A_2$ for the line L_2. If two lines are assumed to be on the ground (i.e., the same U_x), the angle ratio between two associated lines, L_1 and L_2, satisfies:

$$\frac{\tan A_1}{\tan A_2} = \frac{U_{y_2}}{U_{y_1}} \tag{14}$$

where U_{y_1} and U_{y_2} indicate the Y-coordinate values of the lines, L_1 and L_2, respectively. Therefore, the angle ratio, $\frac{A_1}{A_2}$, of two straight lines passing through a vanishing point tells the relative position of the robot with respect to walls. For example, when $\frac{A_1}{A_2} = 1$, the robot is located at the middle of two walls.

If we assume that the line is on the floor, then U_x can be obtained from camera calibration (i.e., U_x corresponds to the height of camera from the floor). Therefore, the relative position of the robot with respect to the wall can be computed by [1]

$$U_y = \frac{U_x}{\tan A_1}. \tag{15}$$

Using Eqs 12 and 15, we can also compute the depth to the points p_1 and p_2 by

$$\begin{aligned} \lambda_1 &= f\frac{U_y}{y_1} \\ \lambda_2 &= f\frac{U_y}{y_2}. \end{aligned} \tag{16}$$

In this section, we have described how vanishing points are related to the transformation between the camera and the world coordinate systems. In practice, it requires two important assumptions:

1. a vanishing point can be computed;

2. two associated lines on the ground floor can be extracted.

In a normal indoor environment consisting of corridors and walls, a vanishing point can be reliably obtained by the vanishing point detection method (Section 2), along with the exploratory motion of the robot - the robot rotates itself until it finds a vanishing point.

Extracting two associated lines on the ground floor can be done by a stereo system. Assume that the robot is aligned with walls by using the extracted vanishing point. We take a pair of images with a stereo system for which the baseline distance, b_d, is specified. From both images, we extract vanishing points and all the associated lines passing through these vanishing points. From Eq. 13, pairs of two corresponding image lines from the left and right images must satisfy:

$$\begin{aligned} \tan A_l &= \frac{U_x}{U_y} \\ \tan A_r &= \frac{U_x}{U_y + b_d} \end{aligned} \tag{17}$$

where A_l and A_r indicate the angle between the associated line and the image y-axis for the left and right image, respectively, and b_d is the baseline distance of a stereo system.

Since A_l, A_r, and b_d can be measured, we can obtain U_x and U_y by:

$$\begin{aligned} U_x &= b_d\frac{\tan A_l \tan A_r}{\tan A_l - \tan A_r} \\ U_y &= b_d\frac{\tan A_r}{\tan A_l - \tan A_r}. \end{aligned} \tag{18}$$

Using these XY coordinates values of parallel lines, we can easily find two associated lines on the floor.

We present experimental results using a stereo system to demonstrate the accuracy of our method. In these experiments, we choose two parallel lines on the ground and measure the distance between the ground and the camera - the correct height. The angles A_l and A_r of two lines are computed by the vanishing points detection method for a pair of stereo images. We can now estimate the X coordinate values,

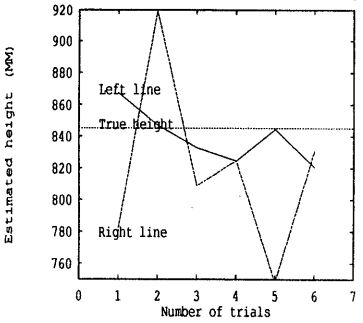


Figure 7: Accuracy of the estimated heights of two associated lines on the ground.

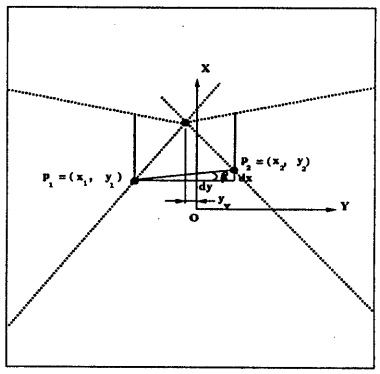


Figure 8: A vanishing point and an intersection.

U_x, of these two lines by Eq. 18. Fig. 7 shows the estimated heights of two parallel lines on the ground floor.

The maximum error, which is smaller than $100mm$, is observed for the right line. The greater errors in the estimated heights of the right line are caused by the inaccurate localization of the image edges for the right line. Even with this maximum error, our current navigation system has successfully guided the robot along a narrow corridor using a vanishing point and two associated lines.

3.2 Intersection-tracking behavior

In an indoor environment, when a behavior-based robot fails to accomplish its mission, such as finding a specific target, it may be a better strategy to find and move to a nearest intersection rather than wandering around.

Unfortunately, finding and tracking intersections is a much more difficult task than wall-following. In the present work, we use a simple heuristic - intersections exist at the end of walls - to extract an intersection. Vertical edges on the walls are candidates of intersections.

Using the vanishing point computed by the algorithm described in Section 2, we can extract and track intersections as follows:

1. First compute a vanishing point;
2. Find two corresponding straight lines on the floor, which form the vanishing point (section 3.1);
3. Extract vertical lines on each straight line, located at approximately the same distance from the robot;

The third step is trivial when the optical axis of camera is aligned with the wall (i.e., when $y_v = 0$). We can just extract two vertical lines intersecting with two wall lines at the same x-coordinates. Even when the optical axis is not aligned with the wall, we can extract two corresponding vertical lines at the same distance from the robot using the position of the vanishing point, y_v, (see Fig. 8).

[1] U_y can be also obtained by a stereo system - by computing the angle change of the associated lines between two stereo images.

Using simple transformations, we can find the relationship between the robot steering angle, θ, and the angle formed by $\mathbf{p_1}$ and $\mathbf{p_2}$, β by

$$\begin{aligned} \tan\beta &= \frac{\Delta x}{\Delta y} \\ &= \frac{C(U_{y_2} - U_{y_1})\sin\theta}{\lambda(U_{y_2} - U_{y_1})} \\ &= \frac{C\sin\theta}{\lambda} \end{aligned} \tag{19}$$

From Equations 12, 19 and 8, we get

$$\begin{aligned} \tan\beta &= \frac{x_1\sin\theta}{f} \\ &= \frac{x_1\sin^2\theta}{y_v\cos\theta} \end{aligned} \tag{20}$$

where f is the focal length of the camera and θ is the steering angle of the robot.

Fig. 9 shows the experimental results of the extraction of an intersection.

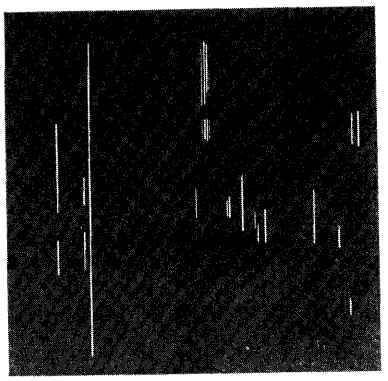
With these extracted vertical edges, intersection-tracker tracks the edges in the following steps:

1. it builds two matching templates by assigning small windows (e.g., 20 × 40) around each vertical line;
2. when a new image is taken, it finds the corresponding vertical lines by computing the cross-correlation between the old and new image (SSDA (Sequential Similarity Detection Algorithm) [1]);
3. it updates matching templates with the pixel values of the new image.

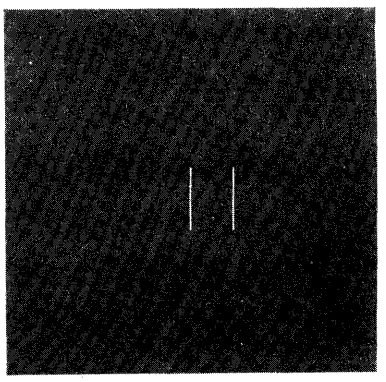
Using the tracking results, intersection-tracker continuously controls the robot so that the two vertical lines always remain around the image center. The steering angle, θ, of the robot to realize this tracking is simply given by:

$$\tan\theta = \frac{\frac{y_2+y_1}{2} - y_c}{f} \tag{21}$$

where y_c is the y-coordinate value of the image center, and y_1 and y_2 represent the y-coordinates of two vertical edges.



(a) Candidate vertical edges



(b) Extracted intersection

Figure 9: Extracted intersection using the vanishing point and edges.

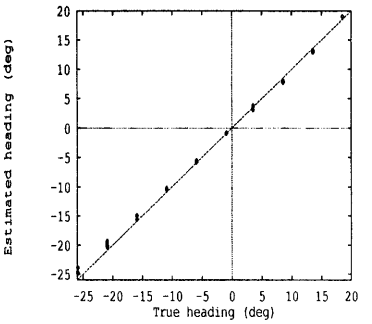


Figure 10: Accuracy of the estimate of the steering angle.

4 Experimental results

One concern in using vanishing points for navigation is the robustness. Errors in vanishing points may produce the erroneous estimate of the robot heading and the distance from the wall. We have done a series of experiments to investigate the robustness issue for both the steering angle and the distance from the wall.

Fig. 10 shows experimental results for the estimation of the steering angle. The diamond marks indicate the estimated steering angle of the robot using a vanishing point (Section 2). The line indicates the correct steering headings which were directly obtained from the robot controller. The largest error in the steering was 3.5^o when the image position of the extracted vanishing point showed 20 pixels error compared to those of accurate vanishing points. In most cases, however, the accuracy of the estimate of the steering angle was satisfactory for navigation purpose.

The true distance from the wall was obtained by a tape measure and Eq. 15 provides an estimate of the distance using an extracted vanishing point. Under ideal conditions, we should observe a linear relationship between the correct distance (or true distance) and the estimated distance. The slope specifies the accuracy of the estimate (e.g., we have the perfect estimate when the slope is equal to 45^o.).

Fig. 11 shows experimental results - the diamond marks indicate the estimated distances from the wall by Eq. 15 and the line has a 45^o slope to show the perfect estimate. While we were arbitrarily changing the vehicle's heading between $\pm 10^o$ [2], the distance from the wall was computed. We can observe good correspondences between the estimate and the correct distances. Even with a very inaccurate estimate of the heading (10^o), the maximum error in the estimate was smaller than $100mm$.

Fig. 12 shows the distribution of the estimates as a function of the robot's heading, at a measured distance of $560mm$. The horizontal line indicates the correct distance, $560mm$. We can observe the maximum error ($100mm$) at the heading, 15^o. Moreover, we found that even when the error in the

[2]Note that Eq. 15 is valid when the heading is equal to zero.

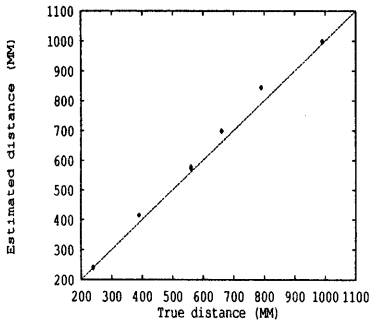


Figure 11: Accuracy of the estimate of the distance from a wall.

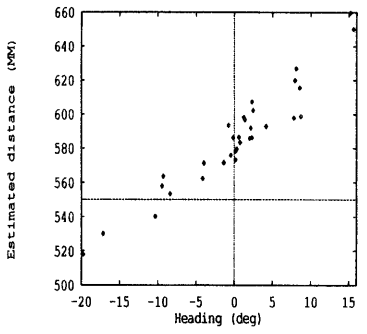


Figure 12: The estimate of distance from a wall as a function of the heading.

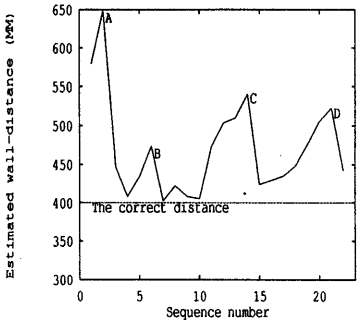


Figure 13: The computed distances from the wall by wall-following behavior.

vanishing point was as large as 10 pixels, the errors in the estimated heading and wall distance were smaller than 2^o and $3cm$, respectively.

We have obtained some preliminary results from a series of navigation experiments in a real office environment, using an indoor mobile robot. In the experiments, we have independently tested two behaviors:

- to follow a wall along the centerline of the corridor (wall-following behavior); and
- to find an intersection and track it (intersection-tracking behavior).

Wall-following behavior worked in the following steps:

1. at first, the robot looks for a vanishing point using the vanishing-point extractor;
2. it then extracts two associated lines on the floor;
3. it computes the slope of the two lines;
4. it computes the distance from the wall, U_{y_1} (Eq. 15);
5. it computes the distance from the other wall, U_{y_2}, using the slopes and U_{y_1} (Eq. 14);
6. it moves to the center position where U_{y_1} is equal to U_{y_2};
7. at this position, it rotates back toward the vanishing point;
8. it repeats the steps from (1) to (7).

Fig. 13 shows the correct distance (i.e., half of the corridor width) and the computed wall distances by wall-following behavior. When the computed distance shows a large discrepancy with respect to the correct distance (e.g., at the points A, B, C, and D in the figure), the behavior controls the robot to the center of the corridor. The robot navigated the distance of travel of 10 meters along a narrow corridor without any failure in a few trials.

In another series of real experiments, intersection-tracking behavior extracts two vertical lines using the vanishing point

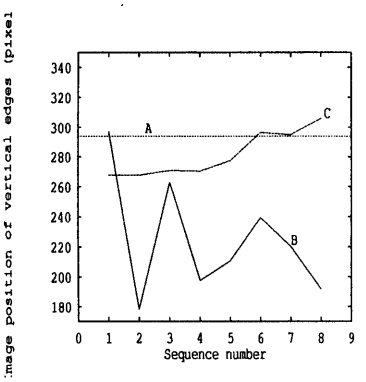


Figure 14: The y-coordinate values of vertical edges, tracked by intersection tracking behavior.

and image edges, and tracks them in real-time (Section 3.2). Fig. 14 shows the computed y-coordinate values of vertical edges from the experiments. The straight line A represents the y-coordinate value of the correct image center. The curve B indicates the center point between two vertical edges before the robot moves. The discrepancies between the center points from the curve B and the correct image center are used by intersection-tracking behavior to control the robot. The curve C shows the resulting center points, corrected by intersection-tracking behavior. Under ideal conditions, we should not observe any errors between the correct image center (the line A) and the corrected center of the vertical edges (the curve C). However, when vertical edges are distant (e.g., $\sim$ 10 meters from the robot) in the beginning of the experiment, the large errors (30 pixels) are observed. As the robot approaches to the vertical edges, the error is reduced to $1 \sim 5$ pixels.

5 Conclusion

We have demonstrated that a navigation system with two vision-based behaviors can provide goal-directed navigation capability for a mobile robot working in indoor environments, consisting of corridors and walls. From our preliminary experimental results, we have found that vision-based behaviors using vanishing points are robust against errors of preprocessing (e.g., edge detection) and image noise.

We have developed a simple but efficient method to extract vanishing points using edge information. Since many parallel lines form one vanishing point, the resulting vanishing point is robust against noise, such as short edge segments and edge localization error.

Only using a vanishing point and the associated edges on the floor, we have demonstrated that a mobile robot can reliably follow a wall. However, one of many drawbacks includes the computational complexity of wall-following behavior. The behavior now controls the robot in the "stop-process-and-move" mode and extracts the vanishing points at every stopping position. In the future, the behavior will extract vanishing points at the starting position and tracks two associated lines using real-time snakes. When real-time snakes cannot track the two lines, the vanishing point extraction method will be invoked.

Extracting and tracking vertical edges, constrained by a vanishing point, has been effective for tracking an intersection. In real experiments, however, the tracking method using SSDA algorithm has failed in many cases due to the sensitivity of image noise. We plan to use real-time snakes to reliably track vertical edges.

We are now developing other vision-based behaviors such as obstacle-avoidance and target-approaching. The obstacle-avoider computes the surface orientation and the time to contact from the image velocity field [4].

References

[1] D. Barnea, , and H. Silverman. A class of algorithm for fast digital image registration. *IEEE Trans.*, 2, 1972.

[2] B. Brillauly-O'Mahony. New method for vanishing point detection. *CVGIP: Image Understanding*, 54, September 1991.

[3] R. A. Brooks. Intelligence without reason. Technical report, Artificial Intelligence Lab., MIT, 1991.

[4] R. Cipolla. *Active Visual Inference of Surface Shape*. PhD thesis, Dept. of Engineering Science, Univ. of Oxford, 1991.

[5] A. Elfes and L. Matthies. Sensor integration for robot navigation: Combining sonar and stereo range data in a grid-based representation. Technical report, The Robotics Institute, CMU, 1987.

[6] E. Gat. Robust low-computation sensor-driven control for task-directed navigation. In *IEEE Robotics and Automation, Scramento*, 1991.

[7] E. Holenstein. Collision avoidance in a behavior-based mobile robot design. In *IEEE Robotics and Automation, Scramento*, 1991.

[8] Kanatani. Reconstruction of consistent shape from inconsistent data. *Int. J. Computer Vision*, 3, March 1989.

[9] I. Kweon and T. Kanade. High resolution terrain map from multiple sensor data. *IEEE Pattern Analysis and Machine Intelligence*, 10, April 1991.

[10] I. Kweon, K. Onoguchi, M. Watanabe, and Y. Kuno. Behavior-based mobile robot using multiple sensors. In *The First Korea-Japan Joint Conference on Computer Vision*, 1991.

[11] M. Magee and J. Aggarwal. Determining vanishing points from perspective images. *Computer Vision Graphics Image Process*, 26, September 1984.

[12] A. Stentz. *The Navlab System for Mobile Robot Navigation*. PhD thesis, School of Computer Science, CMU, 1990.

[13] E. Yamauchi. A behavior-based architecture for robots using real-time vision. In *IEEE Robotics and Automation, Scramento*, 1991.