Steering a Robot with Vanishing Points

Rolf Schuster, Nirwan Ansari, and Ali Bani-Hashemi

Abstract—**The paper analyzes the use of vanishing points for steering a robot. Parallel lines in the environment of the robot are used to compute vanishing points which serve as a reference for guiding the robot. To accomplish the steering task, three subtasks are performed: detection of straight lines, computation of vanishing points, and robot steering using vanishing points. Straight lines are detected by employing a high precision edge detector and a line-fitting algorithm. The cross product method introduced by Magee and Aggarwal is modified to make the detection of vanishing points appropriate for an indoor environment. Properties of vanishing points under camera rotation and translation are derived. Using these properties, the location of the vanishing points can serve as a reference for steering the robot. A model of the robot environment is defined, summarizing the minimum number of constraints necessary for the method to work. Finally, the limitations as well as the advantages of using vanishing points in robot navigation are discussed.**

I. Introduction

This paper analyzes and investigates the usefulness of vanishing points for steering a robot. The objective is to steer a mobile robot based on the vanishing points of parallel lines in its environment. The overall analysis can be described in three sections. Section II briefly describes a method for extracting straight lines in the robot environment. Section III describes the computation of the vanishing points based on the detected lines. In Sections IV and V, positions of the vanishing points are integrated in the steering process of the robot. Furthermore, a model of the robot environment summarizing the essential assumptions which are necessary for the method to work is defined. Note that vanishing points are quite useful to detect accurate robot orientation but they are insensitive to robot translation, and hence the method is not capable of detecting the robot position. Section VI presents experimental results, and concludes with remarks on the usefulness of vanishing points in robot navigation.

II. Edge Detection and Line Fitting

A hierarchical edge detector [1] that combines the first and second derivative operators is used to detect the edges. Both the first and second derivative operators are applied separately to the original image. Then the results are combined by accepting only those zero-crossings produced by the second derivative operator as an edge point, if at the same location, the result of the first derivative operator is above a certain threshold. Thus one can make use of the precision of the second derivative operator (zero-crossings), and "mask out" the high frequency image noise by the first derivative operator. The Sobel and the Laplacian of Gaussian (LoG) operators are used as the first and second derivative operators, respectively. The line fitting algorithm uses recursive subdivision to derive raw lines from the edges. Raw lines are approximated by straight lines using a least squares fit. A detailed discussion on the method can be found in [1]. The resulting image in Fig. 1 (the bottom one) demonstrates clearly how precisely the edge detection algorithm detects various types of edges in the original image.

Manuscript received September 4, 1992; revised March 18, 1993.

R. Schuster is with Siemens AG, ZFE ST SN61, Otto Han Ring 6, N-8000 Munich 83, Germany.

N. Ansari is with the Electrical and Computer Engineering Department, New Jersey Institute of Technology, University Heights, Newark, NJ 07102.

A. R. Bani-Hashemi is with Siemens Corporate Research, Inc, 755 College Road East, Princeton, NJ 07540.

IEEE Log Number 9212614.

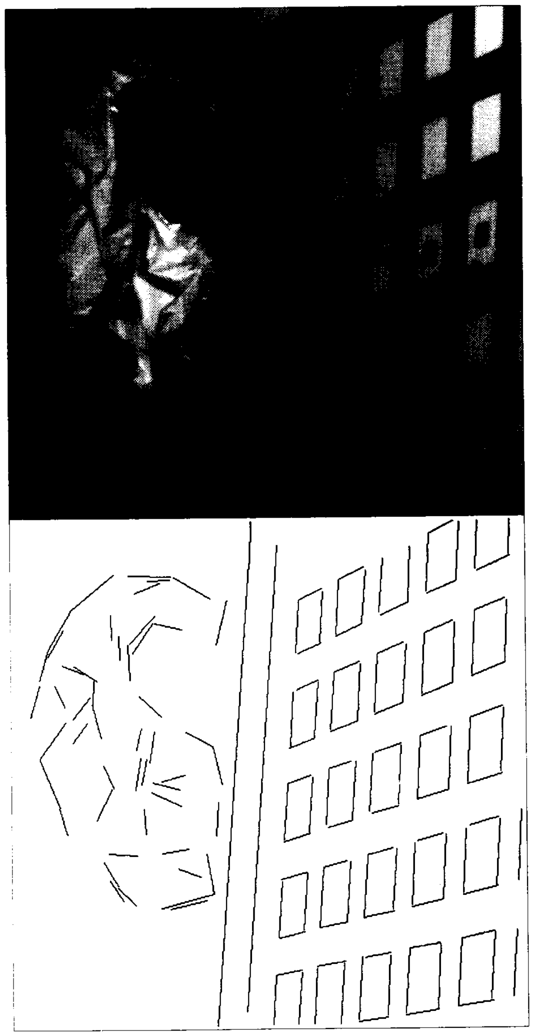

Fig. 1. The top image is the original image, and the bottom image results by applying the edge detection and line fitting algorithm.

III. Detection of Vanishing Points

A fundamental problem in computer vision is how, given a two-dimensional (2-D) image, to derive information about the three-dimensional (3-D) space. One method to derive information about 3-D space from 2-D images is by finding the vanishing points.

A. Concept of Vanishing Points

The concept of vanishing points is closely related to the properties of perspective projection [2]. At this point it is convenient to define the following camera model: the origin of the system coincides with the center of the lens and the z-axis with the optical axis. The image

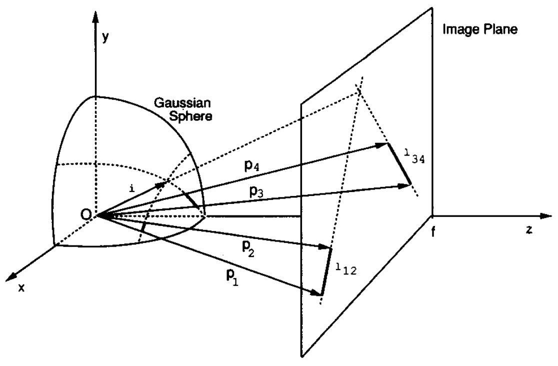

Fig. 2. Gaussian sphere and computation of the vector pointing towards the intersection of two line segments (f denotes the focal length).

plane is defined by the equation $z = f$, where f is the focal length. Hence, the plane is located in front of the center of the lens.

Let l be a straight line through the base point $P_0 = (x_0, y_0, z_0)$ in 3-D space. The equation of l can be written as $\boldsymbol{v} = \boldsymbol{v}_0 + \lambda \boldsymbol{v}_d$, with $\boldsymbol{v}_0 = (x_0, y_0, z_0)^t$ and $\boldsymbol{v}_d = (a, b, c)^t$. Projecting a point P_n which is on the straight line l into the image plane by perspective projection results in a point P_i in the image plane, where $P_i = (x_i, y_i, z_i)$,

$$\begin{aligned} x_i &= \left(f \frac{x_0 + \lambda a}{z_0 + \lambda c}\right) \\ y_i &= \left(f \frac{y_0 + \lambda b}{z_0 + \lambda c}\right) \\ z_i &= f. \end{aligned} \tag{1}$$

As λ approaches infinity, we can neglect $P_0 = (x_0, y_0, z_0)$, and thus P_i approaches the vanishing point V_l. One can define the vanishing point $V_l = (x_l, y_l, z_l)$ of a straight line l as follows:

$$\begin{aligned} x_l &= \lim_{\lambda \to \infty} f\left(\frac{x_0 + \lambda a}{z_0 + \lambda c}\right) = f\left(\frac{a}{c}\right) \\ y_l &= \lim_{\lambda \to \infty} f\left(\frac{y_0 + \lambda b}{z_0 + \lambda c}\right) = f\left(\frac{b}{c}\right) \\ z_l &= f. \end{aligned} \tag{2}$$

Here it is assumed that $c \neq 0$, i.e., the straight line is not parallel to the image plane. If $c = 0$, the vanishing point V_l does not exist (or V_l is infinitely far away in the image plane). Note that the vector pointing in the direction of the vanishing point V_l is parallel to the direction of the straight line l. For later reference it is useful to define two properties.

Property 1 The vanishing point V_l of a straight line l in 3-D space must lie on a line L in the image plane. Any segment of line l, projected onto the image plane, is part of line L.

Property 2 Let Ω be a set of parallel lines in 3-D space. Then all lines of Ω must have the same vanishing point V_Ω.

All these findings about the concept of vanishing points are well understood and are partly described in [2]–[5].

B. Cross-Product Method

There are various methods to determine the vanishing points in a 2-D image [3]–[8]. Some methods are preferable depending on the requirements imposed by the application and the heuristics available from *a priori* knowledge of the 3-D scene. In the following, we describe the cross-product method that has been suggested by Magee and Aggarwal [9], and is based on a previous work by Barnard [4].

The approach uses the Gaussian sphere to represent points in the image plane (see Fig. 2). Any vector to a point in the image plane can be mapped onto the Gaussian sphere (centered at the origin). Any vector from the origin to a point, mapped onto the Gaussian sphere, can be uniquely described in terms of azimuth α (the angle between the projection of the vector onto the xy-plane and the x-axis) and the elevation β (the angle between the vector and the xy-plane).

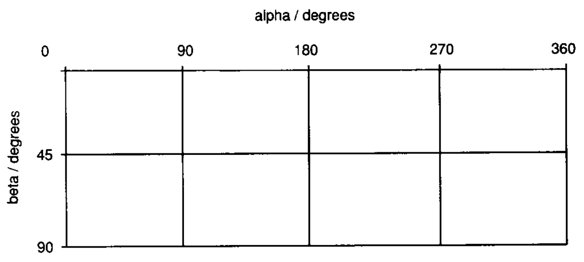

Fig. 3. Two-dimensional accumulator array representing the Gaussian sphere.

Magee and Aggarwal [9] suggest a computationally efficient way to find the intersections of line segments. They find the vector pointing towards the line intersection using three cross products. Considering two line segments l_{12} and l_{34} in the image plane (see Fig. 2), we can define four vectors to the start and end points of the line segments as

$$\boldsymbol{p}_1 = \begin{pmatrix} x_1 \\ y_1 \\ f \end{pmatrix}, \quad \boldsymbol{p}_2 = \begin{pmatrix} x_2 \\ y_2 \\ f \end{pmatrix}$$

$$\boldsymbol{p}_3 = \begin{pmatrix} x_3 \\ y_3 \\ f \end{pmatrix}, \quad \text{and } \boldsymbol{p}_4 = \begin{pmatrix} x_4 \\ y_4 \\ f \end{pmatrix}. \tag{3}$$

The vectors $\boldsymbol{p}_1, \boldsymbol{p}_2$ and $\boldsymbol{p}_3, \boldsymbol{p}_4$ are pointing towards the line segments l_{12} and l_{34}, respectively. Using these definitions, we can find the unit normals to the planes defined by two vectors, i.e., $\boldsymbol{n}_{12}$ and $\boldsymbol{n}_{34}$,

$$\boldsymbol{n}_{12} = \frac{\boldsymbol{p}_1 \times \boldsymbol{p}_2}{|\boldsymbol{p}_1 \times \boldsymbol{p}_2|} \tag{4}$$

$$\boldsymbol{n}_{34} = \frac{\boldsymbol{p}_3 \times \boldsymbol{p}_4}{|\boldsymbol{p}_3 \times \boldsymbol{p}_4|} \tag{5}$$

and thus the vector $\boldsymbol{i}$ along the intersection of the planes,

$$\boldsymbol{i} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \frac{\boldsymbol{n}_{12} \times \boldsymbol{n}_{34}}{|\boldsymbol{n}_{12} \times \boldsymbol{n}_{34}|} = \frac{1}{|\boldsymbol{n}_{12} \times \boldsymbol{n}_{34}|} \begin{pmatrix} \overline{x_i} \\ \overline{y_i} \\ \overline{z_i} \end{pmatrix}. \tag{6}$$

Combining (4), (5) and (6), the components $\overline{x_i}$, $\overline{y_i}$ and $\overline{z_i}$ can be computed by

$$\overline{x_i} = (x_2 - x_1)(x_3 y_4 - x_4 y_3) - (x_4 - x_3)(x_1 y_2 - x_2 y_1) \tag{7}$$

$$\overline{y_i} = (y_3 - y_4)(x_1 y_2 - x_2 y_1) - (y_1 - y_2)(x_3 y_4 - x_4 y_3) \tag{8}$$

$$\overline{z_i} = (x_4 - x_3)(y_1 - y_2) - (x_2 - x_1)(y_3 - y_4). \tag{9}$$

If the line segments or their extensions have an intersection, the vector $\boldsymbol{i}$ is directed towards the point of intersection. If the lines are exactly parallel in the image plane, $\boldsymbol{i} = 0$. Owing to the properties of cross products, vector $\boldsymbol{i}$ might point in the opposite direction (with $z_i < 0$), rather than towards the point of intersection. The algorithm for determining intersections has to consider this case.

Recall that the center of the Gaussian sphere is located at the origin of the coordinate system. The accepted intersection vector $\boldsymbol{i}$ is represented by its azimuth α and elevation β:

$$\alpha = \arctan\left(\frac{y_i}{x_i}\right) \tag{10}$$

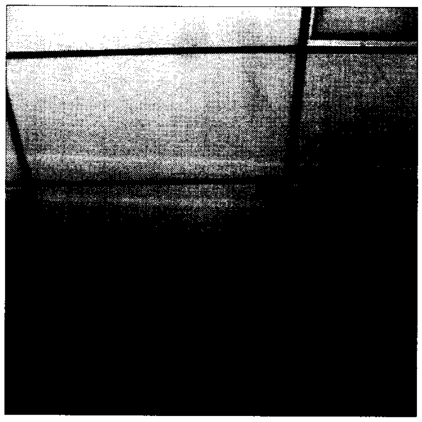

Fig. 4. Image of a ceiling pattern under perspective distortion.

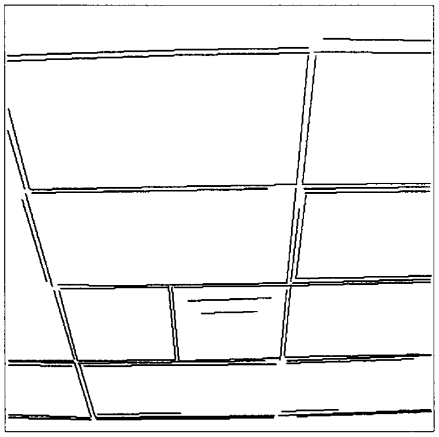

Fig. 5. Line segments in ceiling pattern shown in Fig. 4.

$$\beta = \arctan\left(\frac{z_i}{\sqrt{(x_i)^2 + (y_i)^2}}\right). \tag{11}$$

Since we do not know which line segments form a common vanishing point, we have to compute all intersections between all possible pairs of line segments ($N(N-1)/2$ possible intersections for N line segments). In order to limit the number of intersections, we do not accept a vector $\boldsymbol{i}$ if it points towards the interior of either of the two line segments that are used to compute vector $\boldsymbol{i}$. In general, vanishing points cannot lie between the start and the end-point of a line segment since vanishing points are defined with λ approaching infinity ((2)). After all pairs of line segments have been processed, we have to cluster the intersections, i.e., we have to decide which of the intersections are close enough, such that they could contribute to the same vanishing point.

C. Modified Cross-Product Method

The cross-product method is appropriate for images with strong, distinguished vanishing points and only few accidental intersections which do not belong to any vanishing point. In general, this is not true for real indoor or outdoor images. In particular, in an indoor environment we cannot assume that the vanishing point will be located within a cluster which contains more than a certain number of intersections, and we are confronted with many accidental

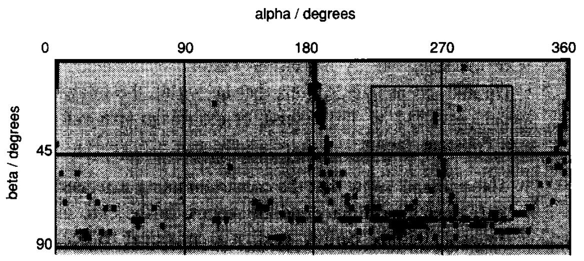

Fig. 6. Accumulator representation of all intersections of the line segments in Fig. 5 (no intersection filtering applied).

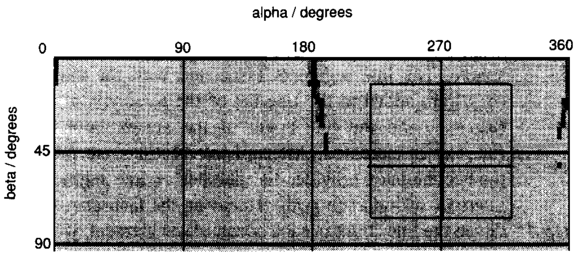

Fig. 7. Accumulator representation of intersections of the line segments in Fig. 5 (with intersection filtering applied).

intersections which do not contribute to a vanishing point. In order to use the cross-product method in an indoor environment with the requirements mentioned above, we modify the following three features of the algorithm: First, additional constraints are used to filter line segments, intersections and (α, β) associations at different stages of the algorithm. Second, a different representation of (α, β) associations and a different clustering algorithm are adopted. Third, a windowing technique based on *a priori* knowledge is used to select the vanishing point from clusters of intersections.

We define a 2-D array (accumulator) with 128 × 32 elements. Each accumulator element represented by (a, b) contains a list of line segments and the count of those line segments. We quantize the occurring (α, β) association in the accumulator such that the values of α (0° to 360°) are mapped into a (128 sections) and the values of β (0° to 90°) are mapped into b (32 sections). See Fig. 3 for illustration. Thus we define a grid on the Gaussian sphere with an angular width and height of 2.81°. This representation of the Gaussian sphere has advantages as well as disadvantages [4], [10]. It is easy to implement and it provides the basis for fast algorithms. The disadvantages are that areas of accumulator elements are not uniform, and a singularity occurs at $b = 32$ ($\beta = 90°$). However, we can still use it for our problem since we are not working close to $\beta = 90°$ (the camera is tilted towards the ceiling).

The modified cross-product method can be summarized in five steps.

Step 1: Compute vector $\boldsymbol{i}$ which points towards the intersection of the line segments using three cross products. Here we add two constraints in order to limit the number of intersections and avoid inaccurate intersections. We consider only those line segments which are longer than a minimum length $l_{\min}$. We do not intersect line segments that are approximately parallel and very close to each other [11].

Step 2: For the indoor environment, the constraint used in the original cross-product method falls short of filtering out the "accidental" intersections (see Fig. 6). Additional constraints are necessary:

- Do not accept intersections that are interior to the *extended* line segments. Line segments can be extended on both sides of the segments by a constant factor.
- Do not accept intersections that lie within the image size (in our case, 512 × 480 pixels), or the image size extended by a constant factor.

Experiments show that the constraint using extended lines is sufficient for ordinary indoor scenes. In the case of images from an indoor ceiling grid, it is practical to accept only those intersections which are outside of the image. Clearly, this constraint is not necessary for the method to work but is included to increase the computational efficiency.

Step 3: Compute the angles α and β of the accepted intersection. Then we find the accumulator grid element (a_n, b_n) corresponding to the intersection (α, β) on the Gaussian sphere. The line segments associated with the intersection (α, β) are added to the line list of the accumulator element (a_n, b_n), and the line count of the accumulator element (a_n, b_n) is increased. After all intersections are processed, the accumulator provides the number of line segments that intersect within each grid element on the Gaussian sphere.

Step 4: Given the line count in each accumulator element, we have to cluster adjacent accumulator elements which have non-zero line count. The clustering algorithm scans the accumulator twice and labels connected elements (8-connected) that have nonzero line count, with the same label.

Step 5: Compute the centers of the intersections within the clusters. For computational efficiency we approximate the center of the intersections by weighed averaging. With the line count as the weight for each accumulator element, the center of a cluster $(\alpha_{\text{center}}, \beta_{\text{center}})$ can be computed by

$$\alpha_{\text{center}} = \frac{360^\circ \sum_{i=1}^{K} n_i a_i}{128 \sum_{i=1}^{K} n_i} \tag{12}$$

$$\beta_{\text{center}} = \frac{90^\circ \sum_{i=1}^{K} n_i b_i}{32 \sum_{i=1}^{K} n_i} \tag{13}$$

where n_i represents the line count, a_i and b_i indicate the position of the element in the accumulator and K is the number of accumulator elements in that cluster. The constant factors only transform the accumulator representation into angles in degrees.

This modified cross-product method provides the direction of all vanishing point candidates uniquely determined by azimuth α_{center} and elevation β_{center}. Now the desired vanishing point has to be selected from the group of candidates. Provided that there exists some *a priori* knowledge of the approximate location of the vanishing point, it is possible to define a window in the accumulator indicating where the vanishing point is expected. Then a vanishing point which has the highest line count within the window is selected.

D. Results

Fig. 4 shows the original image of a room ceiling with a common grid pattern, texture, fans and lamps. The detected lines are shown in Fig. 5. The ceiling grid contains two groups of detected lines (the horizontal and vertical line group). The lines of each group should form one common vanishing point. The modified cross-product method is applied to the detected lines. Fig. 6 shows all the line intersections resulting from Fig. 5 without any filtering constraints. Fig. 7 shows the resulting intersections detected by the modified cross-product method. The gray value of the accumulator elements indicates the number of intersections located within that area of the Gaussian sphere (the darker the more intersections). Three large clusters are detected. The smallest cluster around ($\alpha = 280^\circ, \beta = 53^\circ$) results from the vertical line group. The other two clusters on the right and the left of the first one result from the horizontal line group. Since these lines are almost parallel, their angle of intersection is very small, and thus the error of intersection is very large [11]. The intersections of that line group are not located within one accumulator element only, but they spread over a large region. As β approaches 0° the vanishing point on the image plane approaches infinity. Hence for a useful measurement of the vanishing point β should not be close to 0°.

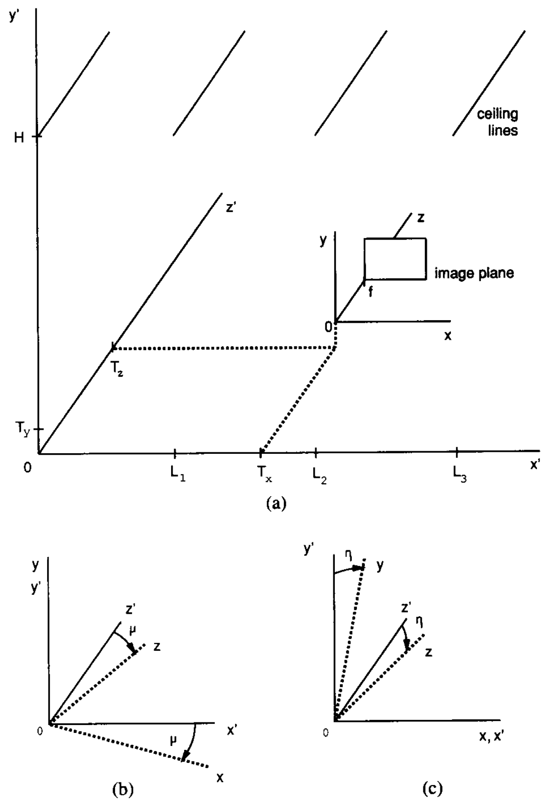

Fig. 8. Coordinate systems. (a) Definition of the room coordinate system and camera coordinate system. (b)–(c) Definition of the angles μ and η.

Since there are more horizontal lines than vertical lines detected, the clusters resulting from the horizontal lines have a higher line count. If we want to select the cluster of the vertical lines as a vanishing point, we have to use a window (provided the approximate location of the vanishing point is known). Fig. 7 shows the window used to select the cluster of the vertical lines as the vanishing point.

The result demonstrates that the modifications made to the original cross-product method are necessary to enable the algorithm to work in an indoor environment. These results illustrate the advantages and limitations of vanishing points as well as the usefulness of the modified cross-product method.

IV. Orientation from Vanishing Points

In general, any approach to robot navigation that uses any type of sensors requires a model of the environment of the robot. In other words, if we want to obtain information about the environment of the robot via sensors, we first have to define that environment. In

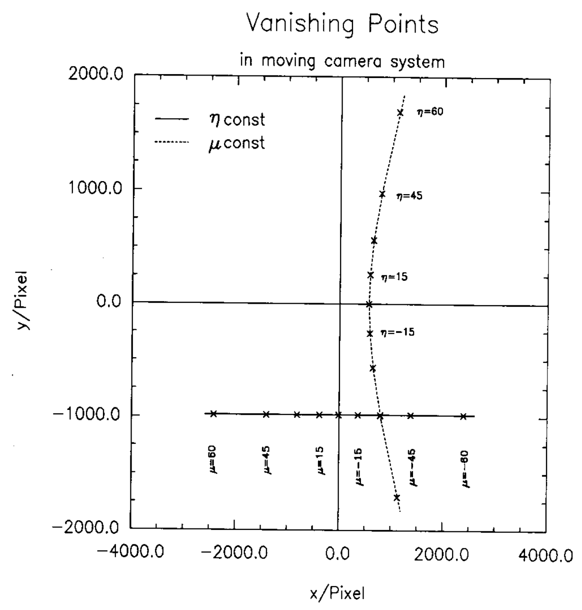

Fig. 9. Movement of vanishing points as a function of the angles μ and η.

the following we define a model for the robot environment which contains a minimum number of constraints necessary for steering a robot based on vanishing points.

A. *Definition of the Generic Model*

Kriegman and Binford [12] suggest the definition of a generic model. A *generic model* is a single model that describes a large class of objects. This implies that the generic model of an environment is as general as possible and uses a minimum number of constraints on the environment. This ensures that the method is applicable to many different environments.

The generic model for steering a robot guided by vanishing points can be defined by three constraints:

- A minimum of two parallel, straight lines are detected by the camera.
- The direction of the lines is known with respect to the robot environment.
- The direction of the lines is not parallel to the image plane of the camera.

This generic model describes any environment which provides parallel, straight lines with any known direction (not parallel to image plane). This covers a wide range of different indoor and outdoor environments. There are no assumptions necessary on the orientation of the surfaces containing the lines (roads, ceilings, floors, walls). Only the direction of the lines with respect to the environment has to be known.

B. *Vanishing Points with Moving Camera*

We can define two coordinate systems (see Fig. 8), one representing a room in an indoor environment (room system, x', y' and z'-axes), and the other representing the space the camera points to (camera system, x, y and z-axes). Let $\boldsymbol{p}'_h = [kx', ky', kz', k]$ be the vector representing the point P' in the room system, and let P be the corresponding point and $\boldsymbol{p}_h = [kx, ky, kz, k]$ be the vector in the camera system. The vectors are given in homogeneous coordinates, where k is an arbitrary nonzero constant. Then P' can be transformed

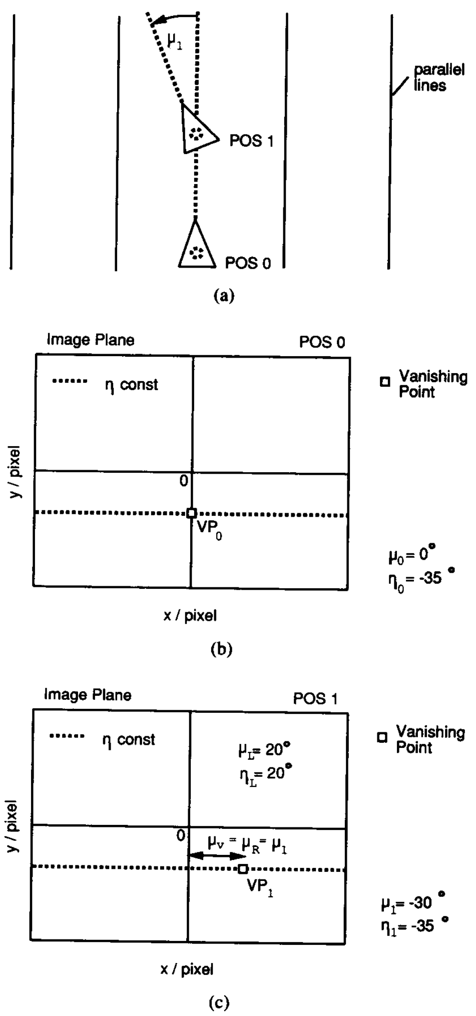

Fig. 10. Vanishing points used for robot steering. (a) Robot in two positions. (b)–(c) Vanishing point in the image plane for robot position 0 and 1.

into P using

$$\boldsymbol{p}_h = p'_h T \mathfrak{R}_\mu \mathfrak{R}_\eta \tag{14}$$

where

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -T_x & -T_y & -T_z & 1 \end{bmatrix} \tag{15}$$

$$\mathfrak{R}_\mu = \begin{bmatrix} \cos\mu & 0 & \sin\mu & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\mu & 0 & \cos\mu & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

$$\mathfrak{R}_\eta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\eta & -\sin\eta & 0 \\ 0 & \sin\eta & \cos\eta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

(see Fig. 8 for the definition of the translation components T_x, T_y, T_z and the angles μ and η).

Equation (14) can easily be used to transform lines from the room system into the camera system. In general, a line, $\boldsymbol{l}' = \boldsymbol{l}'_o + \lambda' \boldsymbol{l}'_d$,

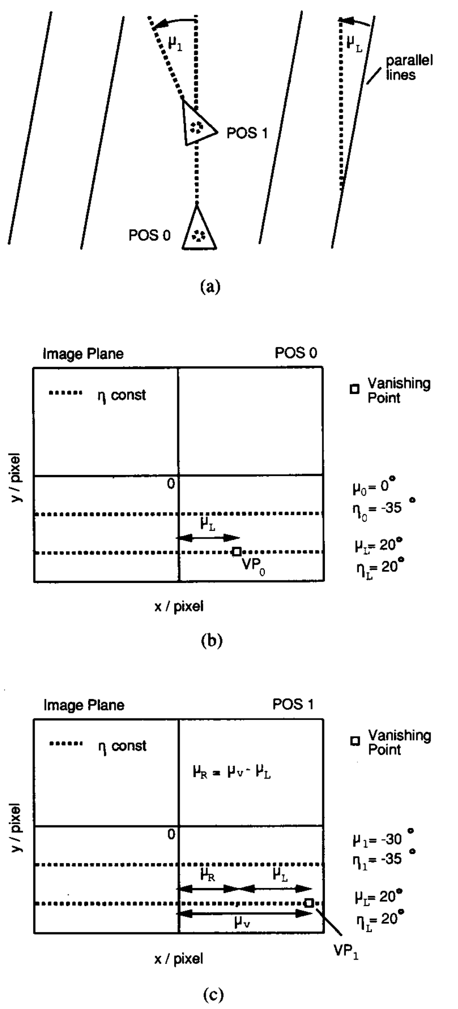

Fig. 11. Vanishing points used for robot steering with rotated and tilted parallel lines (μ_L, η_L). (a) Robot in two positions. (b)–(c) Vanishing point in the image plane for robot position 0 and 1.

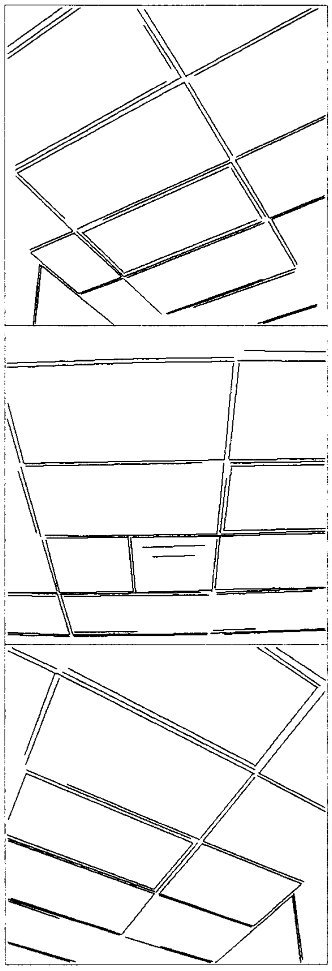

Fig. 12. Detected lines in ceiling pattern with the robot facing three different directions, $\mu = -30°$ (top), $\mu = 0°$ (middle) and $\mu = +30°$ (bottom).

in the room system (with $\boldsymbol{l}_0' = (x_0'y_0'z_0')^t$ and $\boldsymbol{l}_d' = (a'b'c')^t)$ can be transformed into the corresponding line, $\boldsymbol{l} = \boldsymbol{l}_o + \lambda \boldsymbol{l}_d$, in the camera system (with $\boldsymbol{l}_0 = (x_0y_0z_0)^t$ and $\boldsymbol{l}_d = (abc)^t$). To simplify the derivation (without loss of generality), the vector $\boldsymbol{l}_o'$ to the base point of line $\boldsymbol{l}'$ is assumed to be $(L_n, H, 0)$. Note that there are no assumptions made regarding the direction of the line $\boldsymbol{l}'$. Using (14), we transform the vector $\boldsymbol{l}_o'$ (base point) and the direction vector $\boldsymbol{l}_d'$, separately using

$$\boldsymbol{l}_{oh} = \boldsymbol{l}_{oh}' T \mathfrak{R}_\mu \mathfrak{R}_\eta \tag{18}$$

and

$$\boldsymbol{l}_{dh} = \boldsymbol{l}_{dh}' \mathfrak{R}_\mu \mathfrak{R}_\eta \tag{19}$$

where $\boldsymbol{l}_{oh}' = [L_n, H.0, 1]$ and $\boldsymbol{l}_{dh}' = [a', b', c', 1]$. The subscript h indicates homogeneous coordinates. In (19) it is not necessary to apply the translation $\boldsymbol{T}$ since the direction vector is an independent vector, without a specific location associated with it. Equations (18) and (19) determine the parameters of line $\boldsymbol{l}$ in the camera system uniquely. Hence, after converting $\boldsymbol{l}_{oh}$, $\boldsymbol{l}_{dh}$ into Cartesian coordinates, we can write the line equation of $\boldsymbol{l} = \boldsymbol{l}_o + \lambda \boldsymbol{l}_d$ as

$$\boldsymbol{l} = \begin{bmatrix} (L_n - T_x)\cos\mu + T_z \sin\mu \\ (H - T_y)\cos\eta + (L_n - T_x)\sin\mu\sin\eta - T_z\cos\mu\sin\eta \\ (T_y - H)\sin\eta + (L_n - T_x)\sin\mu\cos\eta - T_z\cos\mu\cos\eta \end{bmatrix} + \lambda \begin{bmatrix} a'\cos\mu - c'\sin\mu \\ b'\cos\eta + (a'\sin\mu + c'\cos\mu)\sin\eta \\ -b'\sin\eta + (a'\sin\mu + c'\cos\mu)\cos\eta \end{bmatrix}. \tag{20}$$

This is the transformation of line $\boldsymbol{l}'$ (room system) into line $\boldsymbol{l}$ (camera system) in terms of the location and orientation of the camera in the room system. This enables us to determine the location of the vanishing points as a function of the parameters of the line $\boldsymbol{l}$. Applying (2) on line $\boldsymbol{l}$ given by (20), the vanishing point in the image plane is

$$V_l = f \begin{bmatrix} \dfrac{a'\cos\mu - c'\sin\mu}{-b'\sin\eta + (a'\sin\mu + c'\cos\mu)\cos\eta} \\ \dfrac{b'\cos\eta + (a'\sin\mu + c'\cos\mu)\sin\eta}{-b'\sin\eta + (a'\sin\mu + c'\cos\mu)\cos\eta} \\ 1 \end{bmatrix}. \tag{21}$$

It is assumed that line $\boldsymbol{l}'$ is not parallel to the image plane which implies that the denominators in (21) are not equal to zero. From (21) it is clear that the vanishing point V_l is determined by the direction of the line l', i.e., $l_d' = (a', b', c')$, and the turn and tilt of the camera

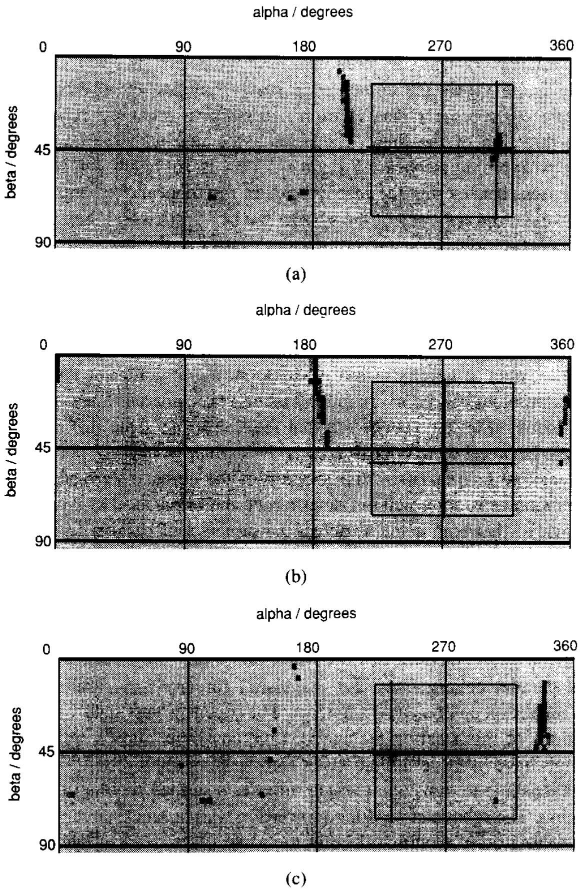

Fig. 13. Detected intersection and vanishing points with the robot facing three different directions, $\mu = -30°$ (a), $\mu = 0°$ (b) and $\mu = +30°$ (c).

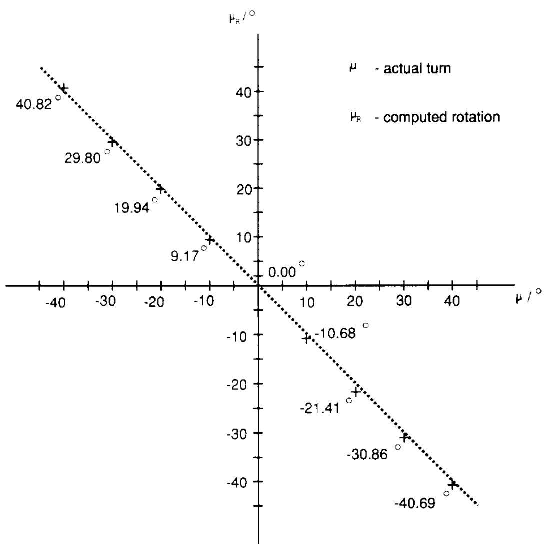

Fig. 14. Experimental results of using vanishing points to determine rotation μ_R for robot turn μ (see text).

in the room system (μ, η). Therefore any line $\boldsymbol{l}'$ with the direction $\boldsymbol{l}'_d$ has the same vanishing point V_l no matter where it is located in the room (Property 2). The vanishing point V_l does not depend on the location of the line $\boldsymbol{l}'$ or the translation (T_x, T_y, T_z). Therefore it is an intrinsic property of vanishing points that they are insensitive to translation [5]. Since there are no assumptions made concerning the orientation of the lines (a', b', c') and the orientation of the camera (μ, η), it is clear that (21) is valid for the general case defined by the generic model.

C. Vanishing Points from Ceiling Lines

To illustrate the properties of generalized vanishing points, (21) is simplified by adding constraints to the generic model. We further assume that the lines are located in the ceiling of the room. In addition, the lines are assumed to be parallel to the z'-axis of the room system (see Fig. 8). Then the line $\boldsymbol{l}' = \boldsymbol{l}'_o + \lambda' \boldsymbol{l}'_d$ can be rewritten as

$$\boldsymbol{l}' = \begin{bmatrix} L_n \\ H \\ 0 \end{bmatrix} + \lambda' \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{22}$$

Using the simplified direction vector $\boldsymbol{l}'_d$ in (21), the vanishing point V_l is defined by

$$v_l = f \begin{bmatrix} \frac{-\tan\mu}{\cos\eta} \\ \tan\eta \\ 1 \end{bmatrix}. \tag{23}$$

Rewriting (23), the components of the vanishing point V_l in the image plane are

$$x_i = \frac{-f\tan\mu}{\cos\eta} \text{ and } y_i = f\tan\eta. \tag{24}$$

Now we can analyze the movement of vanishing points in the image plane for any given movement of the camera (μ, η). Assuming the camera is pointing in the direction of the z-axis $(\mu = \eta = 0)$, the vanishing point is located at the origin of the image plane $(V_l = (0, 0))$. If the camera is rotating (μ variable) with a constant tilt (η constant), the vanishing points form a horizontal line in the image plane (see Fig. 9). For each rotation μ there exists a unique vanishing point on that horizontal line. If the camera tilt is varied (η variable) while there is no rotation (μ constant), the vanishing points are located on a symmetrically shaped curve in the image plane (see Fig. 9). If μ or η approaches 90 degrees, the vanishing point approaches infinity. In any other cases, (24) enables us to predict the location of the vanishing point for any given rotation μ and tilt η of the camera.

V. Steering the Robot

The previous sections assume a rotating camera that is located on the floor (x, z-plane) of the room system. Now the robot is located on the floor and the camera is mounted on the robot such that it is pointing in the direction the robot moves to. The angle μ now represents the turn of the robot, the angle η still represents the tilt of the camera (see Fig. 8).

To illustrate the practical application of the method we first define a model of the robot environment which includes several constraints and assumptions. Later in this section we generalize the environment applying the generic model. The objective here is to steer the robot parallel to the straight lines located on the ceiling of a room. These lines can be some kind of a ceiling pattern or a ceiling grid. The camera is pointing up to the ceiling ($\eta < 90°$) and takes pictures of the ceiling lines. We determine the vanishing point by the modified cross-product method. Knowing (α, β), the location of the vanishing point, we can use

$$x_i = f\frac{\cos\alpha}{\tan\beta} \text{ and } y_i = f\frac{\sin\alpha}{\tan\beta} \tag{25}$$

to find (x_i, y_i) on the image plane. Substituting (25) into (24) and solving for μ and η yields

$$\mu = \arctan\left(\frac{\cos\alpha\cos\eta}{\tan\beta}\right) \text{ and } \eta = \arctan\left(\frac{\sin\alpha}{\tan\beta}\right). \tag{26}$$

These equations can be used to compute the turn and tilt of the robot for any given vanishing point. For steering the robot the tilt of the camera is assumed to be constant. Thus (26) can be used to compute the rotation of the robot necessary to align it with the ceiling lines. Fig. 10(a) shows four ceiling lines and the robot in two positions. In Position 0, it is facing the desired direction, parallel to the ceiling lines ($\mu_0 = 0°$). Therefore, the vanishing point VP_0 is located on the y-axis in the image plane (see Fig. 10(b)). Because of the tilt of the camera ($\eta_0 = -35°$), the vanishing point is located below the origin. In Position 1 the robot is turned to the left ($\mu_0 = 30°$). This results in a vanishing point shifted to the right. Fig. 10(c) illustrates the movement of the vanishing point (VP_1). Based on the vanishing point (VP_1) and the tilt η, (26) can be used to determine μ_R, i.e., the angle the robot is rotated such that it is oriented parallel to the ceiling lines

$$\mu_R = \arctan\left(\frac{\cos\alpha\cos\eta}{\tan\beta}\right). \tag{27}$$

This method of computing the rotation μ_R can be used in a recursive navigation procedure, where the robot repeatedly determines the angle μ_R, corrects its orientation and moves forward.

Now we can generalize the robot environment according to the generic model. A minimum of two parallel lines are detected by the camera. The direction of the lines with respect to the room system is represented by two angles μ_L and η_L, where μ_L denotes the direction of the line with respect to the y', z'-plane and η_L represents the direction of the line with respect to the x', z'-plane. For the generalized environment it is possible to compute the rotation of the robot by

$$\mu_R = \arctan\left(\frac{\cos\alpha\cos\eta}{\tan\beta}\right) - \mu_L \tag{28}$$

in order to turn the robot into the correct direction. Fig. 11 illustrates that concept.

VI. Conclusion

The current implementation of the steering process assumes an environment with lines located in the ceiling of an indoor scene where the robot is supposed to move on the floor in a direction parallel to the lines. Fig. 12 shows images of a ceiling pattern with the robot pointing in three different directions. The line segments have been detected by the line detection algorithm described in Section II. Fig. 4 shows the original image of the ceiling pattern. In the top image of Fig. 12, the robot is turned to the left ($\mu_0 = -30°$). In the middle image, the robot is facing the direction parallel to the lines ($\mu_0 = 0°$). In the bottom image, the robot is turned to the right ($\mu_0 = 30°$). Fig. 13 shows the results of the detected vanishing points corresponding to the three respective directions which the robot points to. The accepted intersections are displayed in the accumulator array representing azimuth and elevation (α, β) on the Gaussian sphere. The vanishing point is selected within the window shown in the accumulator. The location of the vanishing point within the window is marked with a cross. The (α, β) location of the vanishing point is used to compute the angle between the ceiling lines and the direction the robot is pointing to. Experimental results for computing the rotation μ_R from detected vanishing points are shown in Fig. 14. The graph shows the computed μ_R as a function of robot turn μ, where μ_R represents the rotation necessary to point the robot into the direction of the lines. The error associated with the computed μ_R is approximately ± 1.5 degrees. This is sufficient for steering the robot considering the inaccuracies of the robot movements itself. However, the accuracy of the system can be improved (at the expense of computational efficiency) by either reducing the grid size of the accumulator or using the continuous representation for intersections as described in the original cross product method [9].

One limitation of the current system is that detected lines should never be parallel to the image plane. This limits the maximum rotation of the robot to approximately $\pm 90°$ from the direction of the lines. This problem can be overcome by using two cameras. They have to be mounted on the robot such that they have the same tilt, and the angle between their optical axes is $90°$. Since vanishing points are insensitive to translation, the location of the center of projection of the cameras is not crucial in determining the orientation of the robot. Provided that both cameras detect the same set of parallel lines, this setup would enable the system to determine the orientation in any direction (from $0°$ to $360°$ with respect to the lines).

The experimental results demonstrate the usefulness of vanishing points in determining the orientation of a robot. Vanishing points provide a reference for detecting robot rotation. This is true not only for parallel ceiling lines, but in general for any two parallel lines in indoor or outdoor environments. This opens a vast field of applications of this steering technique. Like any other sensor, vanishing points have intrinsic limitations in their sensing capabilities. They are not sensitive to translation which imposes a serious problem in navigation. The robot can be shifted towards a wall or an obstacle, and the vanishing point would still be in the same place. Hence, integrating the vanishing point technique with other sensors are recommended.

References

[1] R. Schuster, N. Ansari and A. Bani-Hashemi, "A hierachical edge detector using the first and second derivative operators," in *SPIE Proc.: Intell. Robots and Comput. Vision XI*, vol. 1824, 1992, pp. 230–241.
[2] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Vol. 2, second ed. New York: Academic, 1982.
[3] S. A. Shafer, T. Kanade and J. R. Kender, "Gradient space under orthography and perspective," in *Proc. Workshop on Comp. Vision: Representation and Control*, 1982, pp. 26–34.
[4] S. T. Barnard, "Methods for interpreting perspective images," *J. Artificial Intell.*, vol. 21, no. 4, pp. 435–462, 1983.
[5] B. Caprile and V. Torre, "Using vanishing points for camera calibration," *Int. J. Comp. Vision*, pp. 127–139, 1990.
[6] J. R. Kender, "Shape from texture," Ph.D. dissertation, Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, 1980.
[7] L. Shigang, S. Tsuji and M. Imai, "Determining of camera rotation from vanishing points of lines on horizontal planes," in *Proc. 3rd Int. Conf. Comp. Vision*, Osaka, 1990, pp. 499–502.
[8] G. Wei and Z. He, "Determining vanishing point and camera parameter: New approaches," in *Proc. 9th Int. Conf. on Pattern Recog.*, 1988, pp. 450–452.
[9] M. J. Magee and J. K. Aggarwal, "Determining vanishing points from perspective images," *Comp. Vision, Graphics, and Image Proc.*, vol. 26, no. 2, pp. 256–67, 1984.
[10] L. Quan and R. Mohr, "Determining perspective structures using hierarchical hough transform," *Pattern Recog. Lett.* , vol. 9, pp. 279–286, May 1989.
[11] H. Nakatani, R. S. Weiss and E. M. Riseman, "An error analysis for surface orientation from vanishing points," in *Proc. SPIE*, vol. 974, pp. 187–194, 1988.
[12] D. J. Kriegman and T. O. Binford, "Generic models for robot navigation," in *Proc. 1988 IEEE Int. Conf. Robotics Automat.*, vol. 2, 1988, pp. 746–751.