Integration of Hough Transform of lines and planes in the framework of conformal geometric algebra for 2D and 3D robot vision

Miguel Bernal-Marin, Eduardo Bayro-Corrochano *  
CINVESTAV, Unidad Guadalajara, Dept. Electrical Engineering and Computer Science, Jalisco, Mexico

A R T I C L E   I N F O
Article history:  
Available online 2 June 2011

Keywords:  
2D and 3D pattern recognition  
Hough Transform  
SLAM  
Geometric algebra  
Robot vision

A B S T R A C T
This paper presents the application of 2D and 3D Hough Transforms together with conformal geometric algebra to build 3D geometric maps using the geometric entities of lines and planes. Among several existing techniques for robot self-localization, a new approach is proposed for map matching in the Hough domain. The geometric Hough representation is formulated in such a way that one can easily relate it to the conformal geometric algebra framework; thus, the detected lines and planes can be used for algebra-of-incidence computations to find geometric constraints, useful when perceiving special configurations in 3D visual space for exploration, navigation, relocation and obstacle avoidance. We believe that this work is very useful for 2D and 3D geometric pattern recognition in robot vision tasks.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Mobile robots are equipped with different input devices to sense their surrounding environment. The laser range-finder is widely used for this task, due to its precision and its wide capture range. Our procedure merges the data obtained by the laser and the stereo camera system to build a 3D virtual map characterized with geometric shapes such as walls and surrounding objects that were obtained by these sensors. This paper is an extension of previous works Bernal-Marin and Bayro-Corrochano (2009a,b).

Using the conformal geometric algebra (CGA), we can represent different geometric entities like lines, planes, circles, and spheres, including the line segments (as a pair-of-points). This framework also allows us to formulate transformations (rotation, translation) using spinors or versors. By using those geometric primitives, we can represent complex 2D and 3D shapes.

There are basically two types of maps suitable for localization: occupancy grid maps and maps based on geometric primitives. This work uses the latter. The knowledge of its pose (position and orientation) while the robot builds a map is a crucial issue for autonomous navigation. Markov Localization (Fox et al., 1999b) is a notable example for a method that do not rely on the extraction of specific features from the sensor data, in which an explicit representation of the probabilistic distribution of the robot’s pose is periodically updated according to the sensor readings, and the Monte Carlo Localization (Fox et al., 1999a) that is a more efficient version of Markov Localization, because it is based on extraction and matching of specific features of only a subset of all possible poses of the robot.

Lines are often used for localization in polygonal environment. For instance in (Gutmann et al., 2001) an algorithm for computing robot poses by line matching is presented. In this work, for the robot localization, we use the line's characteristics in the Hough domain (Hough, 1962) to find the current robot position in the 2D map.

When the environment map has been captured and one starts the mobile robot in a different place, the problem is to relocate itself in the same map. In this work, we use the lines captured in the Hough domain to perform the relocation inside the 2D map. Also Grisetti et al. (2002), present a robot's localization computing the Hough Transform and finding correspondences in a map.

When the relocation task is performed, the mobile robot uses the 3D Hough Transform to represent planes and include them in the 3D map. Our algorithm exploits the 2D and 3D Hough space to localize and find out a particular 3D space configuration, which can be used to recognize spatial areas and objects and navigate safely. We present experiments using real data which validate the efficiency of our approach.

The rest of the paper is organized as: Section 2, we give an outline of the geometric algebra and the conformal geometric algebra. 2D and 3D Hough Transform is presented in Section 3. Section 4 describes the approach for 3D map building. The procedure for robot relocation is explained in Section 5. Section 6 shows a new 3D object representation. Section 7 reports experiment results and Section 8 is devoted to the concluding remarks.

* Corresponding author.
E-mail addresses: mbernal@gdl.cinvestav.mx (M. Bernal-Marin), edb@gdl.cinvestav.mx (E. Bayro-Corrochano).

0167-8655/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2011.05.014
2. Geometric algebra: an outline

The Geometric Algebra (GA) \(\mathcal{G}_{p,q,r}\) is constructed over the vector space \(\mathbb{R}^{p,q,r}\), where \(p\), \(q\), \(r\) denote the signature of the algebra; if \(p \neq 0\) and \(p = r = 0\), the metric is Euclidean; if only \(r = 0\), the metric is pseudo-Euclidean; if \(p \neq 0\), \(q \neq 0\), \(r \neq 0\), the metric is degenerate. The dimension of \(\mathcal{G}_{0,p,q,r}\) is \(2^n\), and \(\mathcal{G}_n\) is constructed by the applications of the geometric product over the vector basis \(e_i\). The geometric product between two vectors \(a, b\) is defined as

\[ ab = a \cdot b + a \wedge b \]

and the two parts; the inner product \(a \cdot b\) is symmetric part, while the wedge product \((outer\ product) a \wedge b\) is the antisymmetric part.

In \(\mathcal{G}_{p,q,r}\) the geometric product of two basis is defined as

\[ e_i e_j = \begin{cases} 1 & \text{if } i = j \in \{1, \ldots, p\}, \\ -1 & \text{if } i = j \in \{p + 1, \ldots, p + q\}, \\ 0 & \text{if } i = j \in \{p + q + 1, \ldots, n\}, \\ e_i e_j & \text{for } i \neq j. \end{cases} \]

This leads to a basis for \(\mathcal{G}_n\) that contains elements of different grades called blades (e.g. scalars, vectors, bivectors, trivectors, etc.):

\[ \{ (e_i)_n \mid \{e_i, e_j, e_k\} \ldots \{e_i, e_j, e_k \ldots e_n\}, \]

which is called a basis blade; where the element of maximum grade is the pseudoscalar \(1 = e_1 \wedge e_2 \wedge \ldots \wedge e_n\). A linear combination of basis blades, all of the same grade \(k\), is called a \(k\)-vector. The linear combination of such \(k\)-vectors is called a multivector, and multivectors with certain characteristics represent different geometric objects or entities (such as points, lines, planes, circles, spheres, etc.), depending on the GA in which we are working (for example, a point \((a,b,c)\) is represented in \(\mathcal{G}_{3,0}\) (the GA of the 3D-Euclidean space \(\mathbb{E}\)) as \(x = ae_0 + be_2 + ce_3\); however, a circle cannot be defined in \(\mathcal{G}_{3,0}\), but it is possible to define it in \(\mathcal{G}_{2,1}\) (CGA) as a 4-vector \(e = e_1 \wedge e_2\) (the intersection of two spheres in the same space). Given a multivector \(M\), if we are interested in extracting only the blades of a given grade, we write \(\langle M \rangle_r\), where \(r\) is the grade of the blades we want to extract (obtaining a homogeneous multivector \(M\) or an \(r\)-vector).

The dual \(X^*\) of an \(r\)-blade \(X\) is defined by \(X^* = X1_n^{-1}\). It follows that the dual of a \((n-r)\)-blade.

The reverse of any multivector \(M\) is defined as

\[ \langle M \rangle_r = (-1)^{nr/2} \langle M \rangle_{n-r}, \quad \text{for } M \in \mathcal{G}_n, \quad 0 \leq r \leq n. \]

The reader should consult Bayro-Corrochano (2010) for a detailed explanation about CGA and its applications.

2.1. Conformal geometric algebra

When working in conformal geometric algebra (CGA), \(\mathcal{G}_{4,1,0}\) means to embed the Euclidean space in a higher-dimensional space with two extra basis vectors which have a particular meaning: in this way we represent particular entities of the Euclidean space with subspaces of the conformal space. The extra vectors we add are \(e_s\) and \(e_\infty\), defined by the properties \(e_s^2 = 1\), \(e_\infty^2 = -1\), \(e_s e_\infty = 0\). With these two null vectors, we define the null vectors \(e_0 = e_1 e_2 e_3\) and \(e = e_s e_\infty\) interpreted as the origin and the point at infinity, respectively. From now on, points in the 3D-Euclidean space are represented in lowercase, while conformal points appear with underlined letters; also, the conformal entities will be expressed in the Outer Product Null Space (OPNS) (notated with an asterisk; also known as the dual of the entity), and not in the Inner Product Null Space (IPNS) (without an asterisk) unless it is specified explicitly. To go from OPNS to IPNS, we need to multiply the entity by the pseudoscalar. To map a point \(x \in \mathbb{E}^3\) to the conformal space in \(\mathcal{G}_{4,1,0}\) (using IPNS), we use

\[ X = X + \frac{1}{2} X e^s + e_\infty. \]

Applying the wedge operator “\(\wedge\)” on points, we can express new entities in CGA. All geometric entities from CGA are shown in Table 1 for quick reference.

<table>
<thead>
<tr>
<th>Entity</th>
<th>IPNS</th>
<th>OPNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>(p - p + (p - p^2) x + e_0)</td>
<td>(p = \mathbb{E} \wedge b \wedge c \wedge d)</td>
</tr>
<tr>
<td>Point</td>
<td>(x = x \wedge e_0)</td>
<td>(x = (\mathbb{E} - \varnothing e_0 + e))</td>
</tr>
<tr>
<td>Plane</td>
<td>(p = N + d)</td>
<td>(p = e \wedge d \wedge b \wedge c)</td>
</tr>
<tr>
<td>Line</td>
<td>(l = e_1 \wedge M_1 + M_1)</td>
<td>(l = e \wedge b \wedge c)</td>
</tr>
<tr>
<td>Circle</td>
<td>(l = l_1 \wedge M_1 + M_1)</td>
<td>(l = e \wedge b \wedge c)</td>
</tr>
<tr>
<td>P-pair</td>
<td>(P = l_1 \wedge M_1 + M_1)</td>
<td>(P = e \wedge b \wedge c)</td>
</tr>
</tbody>
</table>

\[ X = X + \frac{1}{2} X e^s + e_\infty. \]

\[ X = X + \frac{1}{2} X e^s + e_\infty. \]

\[ X = X + \frac{1}{2} X e^s + e_\infty. \]

\[ X = X + \frac{1}{2} X e^s + e_\infty. \]
equations and for linear transformations one uses matrix representation with redundant coefficients. In contrast, conformal geometric algebra, a coordinate-free system, provides a fruitful description language to represent primitives and constraints in any dimension, and by using successive reflections with bivectors, one builds versors to carry out linear transformations avoiding redundant coefficients.

3. Hough transform: overview

The Hough Transform (Hough, 1962) is an effective method to identify the location and orientation of lines (Duda and Hart, 1972). Next we describe the 2D, 3D and the relation of Hough space and Conformal Geometry Algebra.

3.1. 2D Hough transform

The Hough Transform is the mapping of a line \( L \) of \( \mathbb{R}^2 \) into a single point in the 2D Hough space \((h, q)\), where \( q \in \mathbb{R} \) and \( h \in [0, 2\pi) \). The equation describing a line \( L \) in \( \mathbb{R}^2 \) is given by

\[
L : y = ax + b,
\]

where the scalars \( a, b \) are the line parameters. This line \( L \) can then be represented in the parameter space as a point with coordinates \((a, b)\). In an opposite way, a point \((x_i, y_i)\) in \( \mathbb{R}^2 \) can be represented in the parameter space as a line \( L^0 \) as follows:

\[
L^0 : b = -x_ia + y_i,
\]

where \( x_i, y_i \in \mathbb{R} \) are the parameters of the line \( L^0 \). Consider \( p_1, p_2, \ldots, p_n \), a set of points lying on \( L \). Each of these points represents a line in the parameter space. Thus, the intersection of these lines in the parameter space corresponds to the point \( a, b \), which in turn represents the parameters of the line \( L \) in the \( \mathbb{R}^2 \). For the case of \( x = \text{constant} \) in Eq. (7), the \( y \)-axis coefficient equals 0. One way to go around this problem is to resort to the normal form of the line equation: \( L : \cos(\theta)x + \sin(\theta)y = \rho \), where \( \theta \) and \( \rho \) are the parameters of the normal to the line passing through the origin in \( \mathbb{R}^2 \); see Fig. 1a. Thus, one point \((x_i, y_i)\) in \( \mathbb{R}^2 \) represents a sinusoid in the parameter space; see Fig. 1a. The points \((x_i, y_i)\) lying on a line \( L \) in \( \mathbb{R}^2 \) represent a sinusoid, which intersect in just one point \((\theta, \rho)\) in the parameter space. This point represents the line \( L : \cos(\theta)x + \sin(\theta)y = \rho \).

So far, we have defined a function that maps a line in \( \mathbb{R}^2 \) to a point in the Hough space. Let be \( L \in \mathbb{R}^2 \) as defined in (6), the Hough Transform function can be defined as \( HT(L) \rightarrow (\theta, \rho) \). An interesting property (Locchi et al., 2001), that will be useful in the following sections, is in the relation between the transformations of lines when the robot moves.

Property 1. Given the Hough Transform of a set of lines \( L \), sensed by the mobile robot, and a rotation/translation \((T_x, T_y, h_R)\) of the robot (we assume \( h_R \in [0, 2\pi) \)), the Hough Transform of the \( L \) with respect to the new pose of the robot will be \( HT(L) \rightarrow (\theta', \rho') \) such that:

\[
\cos(\phi) \cos(\psi) x + \cos(\phi) \sin(\psi) y + \sin(\phi) z = \rho
\]

where \( \phi, \psi \) are the Euler angles of the robot.

**Fig. 1.** 2D and 3D Hough Transform representation.
if $0 \leq \theta + \theta_k < 2\pi$, 
\[
\rho' = \rho + T_x \cos(\theta + \theta_k) + T_y \sin(\theta + \theta_k),
\]
if $\theta + \theta_k > 2\pi$, 
\[
\rho' = \rho + \theta_k - 2\pi,
\]
\[
\rho' = -(\rho + T_x \cos(\theta + \theta_k) + T_y \sin(\theta + \theta_k)).
\]
If $\theta + \theta_k < 0$, 
\[
\rho' = \rho + \theta_k + 2\pi,
\]
\[
\rho' = -(\rho + T_x \cos(\theta + \theta_k) + T_y \sin(\theta + \theta_k)).
\]

3.2. 3D Hough transform

Extending the principle of the line, we can represent a plane embedded in $\mathbb{R}^3$ as a point $(a,b,c)$ in the 3D parameter space
\[
\Pi : z = ax + by + c.
\]
Similarly, if the equation has the following form:
\[
ax + by + c = dz = 0,
\]
it cannot be represented in the new parameter space as the coefficient $d$ if the $z$-axis is equal to 0. Thus, we again use the normal form of the plane:
\[
\Pi : \cos(\theta)\cos(\phi)x + \sin(\theta)\cos(\phi)y + \sin(\phi)z = \rho,
\]
where $\theta, \phi, \rho$ stand for the parameters of the normal plane passing through the origin; see Fig. 1b. Note that one point $(x_i, y_i, z_i) \in \mathbb{R}^3$ represents a sinusoidal surface in the parameter space $(\theta, \phi, \rho)$. The points $(x_i, y_i, z_i)$ lying on the plane in $\mathbb{R}^3$ represent sinusoidal surfaces that intersect in just one point $(\theta_i, \phi_i, \rho_i)$, which in turn represents the original plane $\Pi$ (Eq. (10)), see Fig. 1d.

3.3. Hough transform and CGA

The 2D Hough Transform maps the $\mathbb{R}^2$ to the 2D Hough space parameterized by $(\theta, \rho)$. Because, in this work, we represent geometric entities in conformal geometric algebra, we need to reformulate the Hough Transform so that we can comfortably use this formulation while we are computing in geometric algebra. Lines in 2D or planes in 3D are in fact hyperplanes embedded in a 2D or 3D linear space; thus, if the Hough Transform was developed for lines, from the mathematical point of view, it should be possible to generalize it for any hyperplane of $\mathbb{R}^n$, where $n > 2$. However, the way we represent such hyperplanes is the key; thus, we resort to formulating a generalized Hough Transform related to conformal geometric algebra. There are just two intrinsic characteristics of hyperplanes: the Hesse distance (a scalar) and the hyperplane orientation (n unit vector). Thus, a hyperplane in $\mathbb{R}^n$ has $n + 1$ degrees of freedom (DOF) and is given by the generalization of Eq. (8):
\[
\omega x_1 + \omega x_2 + \cdots + \omega x_n = \mathbf{a} \cdot \mathbf{x} = \mathbf{a}^\top \mathbf{x} = -z,
\]
where $\mathbf{a} \in \mathbb{R}^n$. This equation can be conveniently rewritten using homogeneous coordinates and a normalization scalar $|\mathbf{a}|$ as follows:
\[
[\mathbf{x}, 1]^\top [\mathbf{a}, 1] = [\mathbf{x}, 1]^\top [\mathbf{n}, h] = \mathbf{x}^\top \mathbf{n} = 0.
\]
This represents the constraint of a homogeneous point $\mathbf{X} \in \mathbb{R}^{n+1}$ lying on the plane $\Pi = [\mathbf{n}, h]^\top$, where $\mathbf{n} \in \mathbb{R}^n$ and $h \in \mathbb{R}$ are the normal and the Hesse distance, respectively.
current reading, and other with global coordinates "g" (according to the initial position of exploration). The use of the encoders helps us to estimate the actual position of the mobile robot, but these readings has problems of inaccurate due to friction wheels. Therefore, the pose of the robot, its rotation angle and translation, are calculated by

\[ \theta = \theta_b + \theta_{\text{error}}, \]
\[ T = T_b + T_{\text{error}}. \]

where \( \theta_b \) and \( T_b \) are the rotation angle and the translation vector given by the odometer, and \( \theta_{\text{error}} \) and \( T_{\text{error}} \) are the value of the correction error generated by the comparison of the actual laser reading (line segments in local map) and the prior reading (line segments in global map). Using a perpendicular line to \((x, y)\)-plane as the rotation axis and using (13) as the angle, and adding a third fixed coordinate to (14), we can apply these values in (3) to get the translator \( T_{\text{pos}} \) and the rotor \( R_{\text{pos}} \), which represent the robot movement.

4.1. Line detection

To extract line segments from laser range points, we use a recursive line-splitting method, as shown in (Zhang and Ghosh, 2000), instead of Hough Transform, because this is a speedy and correctness algorithm that performs divide-and-conquer algorithm (Nguyen et al., 2007). We map every endpoint of the line segments to CGA to get the pair-of-points \((PP^p)\) entity and store it in a local map \( L \). As the endpoints are 2D points, we take the last coordinate \((x, y, z)\)-plane as the rotation axis, and then the form \( x = M_{l0} M_{l1} \)

\[ (s_1 \land s_2)^3 \begin{cases} < 0 & \text{if } s_1 \text{ and } s_2 \text{ intersect}, \\ = 0 & \text{if } s_1 \text{ and } s_2 \text{ are tangent}, \\ > 0 & \text{if } s_1 \text{ and } s_2 \text{ do not intersect}. \end{cases} \]

To get a sphere from a pair-of-points, we use the following equation

\[ S_{PP} = \frac{PP^p}{PP^p \land e}. \]

The possible combination of states of the line segments are:

a. Both spheres intersect and have the same direction.
b. One sphere is inside of the other but has the same direction.
c. Both spheres intersect but the segments do not have the same direction.
d. One sphere is inside of the other but the segments do not have the same direction.
e. Both spheres do not intersect.

These combination are represented in Fig. 3.

When we have obtained the line matching, we merge both maps and correct the angle and displacement of the lines, comparing the local and global maps; this little error is caused by the odometer sensor. Then we update the actual position of the robot using (13) and (14).

We can express a motor that maps any entity that has been taken from the laser coordinate system to the global coordinate system. Taking the laser’s center of coordinates, and making a motor \( M_{\text{lor}} \) that represents the rotation and translation from the center of the global coordinate system to the laser’s center, (see Fig. 4) we compute

\[ M_{\text{cf}} = R_{\text{pos}} M_{\text{lor}} R_{\text{pos}}, \]
\[ M_{\text{pos}} = T_{\text{pos}} M_{\text{lor}}, \]
\[ M_{\text{lo}} = M_{\text{lor}} M_{\text{pos}}. \]

where (17) is the translation and rotation motor toward the laser’s center, (18) is the movement of robot using the laser range-finder, and (19) is the motor that leads to the source of the laser sensor in the global coordinate system. Using (19) with any geometric entity (point, line, circle) recorded with the laser range-finder sensor, we can move it easily to the global coordinate system using then the form \( y = M_{lo} x \).

As we are dealing in a 3D real world and the laser range-finder only shows a plane measure, we can add a virtual wall (Fig. 7a) to the shapes from the laser range-finder to get a 3D visual sense of the walls that are inside the virtual world. If a new laser range-finder is mounted on the mobile robot or if the laser range-finder is moved to another place in the mobile robot, it is easy to get the new motor that maps the data from the laser range-finder to the global map, only updating the motor \( M_{\text{lor}} \) that represents the rotation and translation from the center of the global coordinate system to the laser’s center, and finally recalculating (19).

4.2. Stereo camera system

The pan-tilt unit has two degrees of freedom which can be expressed as two rotation, one for pan and other for tilt. This rotation we can modeled using rotors as (3). Let \( R_{\text{pan}} \) be the rotor for the pan movement and let \( R_{\text{tilt}} \) for the tilt movement. Applying this ro-

Fig. 3. Matching and merging line segments (par-of-point) using spheres.

Fig. 4. Laser coordinate system to global coordinate system.
tors using the geometric product we can model all the pant-tilt system. We apply the method of hand-eye calibration (Bayro-Corrochano et al., 2000), to get the axis from the pan-tilt unit and to compute its intersection (or the closest point between the rotation axis), which help us to build a translation from this intersection to the source of the stereo camera system ($T_{eye}$ translator). Then, we develop a motor that maps any entities taken from the stereo camera system to the global coordinate system:

$$T_{u} = R_p u R_{x} R_{y} R_{z},$$  \hspace{1cm} (20)

$$R_p = R_{x} R_{y} R_{z},$$ \hspace{1cm} (21)

$$T_{opt} = R_p T_{eye} R_{pt},$$ \hspace{1cm} (22)

$$M_{mp} = T_{mp} T_{opt},$$ \hspace{1cm} (23)

$$M_{mp} = T_{mp} T_{opt} R_{mp},$$ \hspace{1cm} (24)

where (20) is the translation to the point that has the minimum distance to the axis of pan-tilt, taking into account rotation of the robot position; (21) is the rotor resulting of all the spins that has done so much in the position of the robot, as in the pan-tilt; (22) is the translation to the left camera of the stereo camera system taking into account all the movements that had the system; (23) is the movements motor of the robot, along with the pan-tilt; (24) is the complete movement motor of the robot. All these relations are depicted in Fig. 5.

Any point captured by the cameras in any angle of the pan-tilt unit, in any position of the robot can be mapped from the stereo camera system and then take them to the global coordinate system (pair-of-points), lines, circles, planes, spheres in the frame of stereo representative points we can represent points, line segments where (20) is the translation to the point that has the minimum distance to the axis of pan-tilt, taking into account rotation of the robot position; (21) is the rotor resulting of all the spins that has done so much in the position of the robot, as in the pan-tilt; (22) is the translation to the left camera of the stereo camera system taking into account all the movements that had the system; (23) is the movements motor of the robot, along with the pan-tilt; (24) is the complete movement motor of the robot. All these relations are depicted in Fig. 5.

Any point captured by the cameras in any angle of the pan-tilt unit, in any position of the robot can be mapped from the stereo camera system to the global coordinate system using the form

$$\hat{x} = M_{mp} M_{mp}^{-1}. \hspace{1cm} (25)$$

Using the CGA we can capture all the entities showed in the Table 1, using the OPNS form. By capturing the 3D objects using its representative points we can represent points, line segments (pair-of-points), lines, circles, planes, spheres in the frame of stereo camera system and then take them to the global coordinate system using (25).

5. Robot relocation on a map using lines

According to Section 3.1, the Hough Transform is the parameterization of a line from the $(x,y)$-plane (a Cartesian plane) to the $(\theta, \rho)$-plane (the Hough domain). The line segments from the map are transformed to the Hough domain, defining the transformation in the domain of $\theta \in [0, 2\pi)$, so every line segment L in $(x, y)$ corresponds to a point $(\theta, \rho)$. If only the angle $\theta$ varies, the value of $\rho$ keeps constant. So, given a previously captured map $\mathcal{G}$ (global map) in terms of 2D lines and a new captured map $\mathcal{L}$ (new local map), the difference between them is an angle $\Delta \theta$ and a displacement $\Delta x$ and $\Delta y$, which affects the $\Delta \rho$ value. Fig. 6 depicts lines from Global map $(a, b, c)$ and Local map $(a', b', c')$, and their Hough representation. Using the Property 1 of Hough Transform, the difference of an angle in the Hough domain is defined as follows:

$$\Delta \theta(\theta_a, \theta_b) = \begin{cases} 
\theta_a - \theta_b - 2\pi & \text{if case 1}, \\
\theta_a - \theta_b + 2\pi & \text{if case 2}, \\
\theta_a - \theta_b & \text{otherwise}.
\end{cases} \hspace{1cm} (26)$$

where case 1: if $\theta_a > \theta_b$ and $\theta_a + \theta_b > 2\pi$ and $\theta_a + \theta_b < 0$; case 2: if $\theta_a < \theta_b$ and $\theta_a + \theta_b > 2\pi$ and $\theta_a + \theta_b < 0$. This gives us the calculation of any point near others, where its angles are close to $0 = 2\pi$. Let $\mathcal{G}^{HT}$ be the Global map using only the lines in the Hough domain.

The relocation follows the next steps:

- **Extract the actual environment**: using the laser range-finder, extract the line segment, map them to the Hough domain, and store it in $\mathcal{L}^{HT}$ ($\mathcal{L}^{HT}$ only has $(\theta, \rho)$ from each line).
- **Take the difference**: for each element in $\mathcal{L}^{HT}$ with each element in $\mathcal{G}^{HT}$ (using (26) in angles) and store it in $\Delta(\theta, \rho)$. This gives us a twist and displacement in Hough domain.

$$\Delta(\theta, \rho) = \mathcal{G}^{HT} - \mathcal{L}^{HT}. \hspace{1cm} (27)$$

This step can be seen as the difference between the actual map and the previously captured map.

- **Build a new global map**: adding all the elements of $\Delta(\theta, \rho)$ to an element $l_i \in \mathcal{L}^{HT}$ and storing it in $\mathcal{G}^{HT}$ as shown in (28):

$$\mathcal{G}^{HT} = \mathcal{G}^{HT} + \mathcal{L}^{HT}. \hspace{1cm} (28)$$

This gives us a displacement of the actual map, close to the global map.

- **Get an error of displacement $\xi$:** the angle $\Delta \theta$ has been shifted in $\mathcal{G}^{HT}$, so we decrease $\mathcal{G}^{HT}$ by the value of $\mathcal{G}^{HT}$, which is its $\Delta \rho$. The goal is to reduce this error using

$$\sum_i \xi_i = \mathcal{G}^{HT} - \mathcal{L}^{HT} = 0. \hspace{1cm} (29)$$

- **Populate the vote matrix $V$:** Let $V$ be a zero-vote matrix of dimension $|\mathcal{G}^{HT}| \times |\mathcal{L}^{HT}|$, whose votes are given if the error of the displacement is less than a threshold:

$$\xi_i < (\xi_0, \xi_\rho), \hspace{1cm} (30)$$

where $\xi_0$ and $\xi_\rho$ are the thresholds from the angle and $\rho$, respectively.

Fig. 6. Map matching for location in Hough domain.
Repeat the last three steps for each line in $\mathcal{L}^{\kappa T}$. Finally, when all the lines have been displaced and voted, we take the maximum value per column from $V$, where the row position of it, corresponds to a line in $\varphi^{\kappa T}$. This is the line correspondence, if the value is null, there is no matching.

Now, we get all the information about which lines are matching, and with this data we can move and rotate the robot to the right place on the map. To compute the rotation angle, we use the angle of matching lines. Then, with it, build a new rotor to turn the robot and the local map $\mathcal{L}$ (line segments) to the new position in global map. So far, we have the orientation of the mobile robot and only remains to know the displacement position. We can find the displacement $\Delta x$ and $\Delta y$ using the closest point to the origin in the matching lines, and generate a translation vector $t$. The closest point to the origin on a line in CGA can be calculated by

$$t = -\left(L \cdot \Xi \right) \cdot \left(\left(e_\varphi \cdot L \right) \Gamma'\right).$$

(31)

As we get line segments (pair-of-points in CGA), we only need to apply the wedge operator with the point at infinity, as shown in (32):

$$L' = PP \wedge e$$

(32)

to get the line in CGA and perform (31). With this translation vector $t$, we make a translator and apply it to the local map and the mobile robot. Finally, the robot is located in the correct place on the map, so we can continue with navigation within the environment. In Fig. 7b, we can see the relocation evolution, where (a) shows the robot initial position, which is taking a sample of the environment, (b) generates the line segments of the actual environment, (c) loads a previous map to perform matching (here we can see that the mobile robot is displaced and turned in a random place), and (d) locate and put the robot in the correct place on the map (here the robot has located itself in the previous environment and is placed in the right place).

Fig. 7. Building 3D maps and performing relocation in a room.

Fig. 8. A mobile robot looking at a workplace and their representation in Hough space (second column) and its variant angle features (third column).
5.1. Computational complexity

The complexity of the computing of our relocation method taken from each step is next explained: the extraction of the actual environment is performed by split-and-merge algorithm that has a complexity of $O(|S|)$, where $S$ is the vector of sensed points in the laser range-finder. Consequently the transformation involvement of these lines to Hough domain represents a complexity of $O(|L|)$. Take the difference of the maps, build a new global map and get the error of displacement represents a complexity of $O(|L_{HT}|^2 + |S|)$. In the worst-case our method reaches a complexity of $O(|G_{HT}|^2 + |L_{HT}|^2 + |S|)$.

The computational complexity of our work may be compared with a similar approach of global localization using line matching described in (Gutmann et al., 2001), that has the worst-case complexity for line matching as $O(|G|^3 |L|^2)$. As far as the line extraction concerns, the use of Hough Transform demands an additional complexity of only $O(|S|)$, that is the case of Grisetti et al. (2002) that has complexity of $O(|G|^3 |L|^2 + |S|)$. Since other methods use point cluster instead of lines, such methods demand a higher computational time for the case of multiple lines processing.

Note that, we are not extracting the lines from using the Hough Transform method, we use the lines already extracted and then transform them to Hough space. Therefore working in the Hough space gives a substantial computation time advantage with respect to other approaches working in the map or robot’s coordinate space.

6. 3D objects representation using Hough domain

As mentioned in Section 4.2, points obtained from stereo images can be used to calculate a representation of a plane in the virtual world. Several techniques can be used to get features from images (Rosten et al., 2010). Plane parameters can be obtained using stereo vision and implementing the RANSAC algorithm (Tarsha-Kurdi et al., 2007). The plane parameters are the radius or radial distance $\rho$ (Hesse distance), the plane tilt (or polar angle) $\theta$, and the azimuth (or azimuthal angle) $\varphi$ (see Fig. 1c). In the environment, the plane representation is displayed using the points lying on it. Three of these points are enough to represent the plane. In the Hough domain $\rho$, $\theta$ and $\varphi$ are used as points in this space as depicted in Fig. 8. Note that a plane is represented by just 3 DOF. Using these points, it is easier to handle planes due to their values ($\rho > 0$, $\theta \in [0, \pi)$ and $\varphi \in (0, 2\pi]$). Also when the mobile robot navigate, only the value of $\rho$ and $\varphi$ are changed.

6.1. Environmental perception

This section presents the application of the 3D Hough Transform to store complex 3D shapes. As far as concerns about the object representations complexity, a natural question arises: which

Fig. 9. A stairway representation and their Hough Transform viewed from different angles (third column $\varphi$ affected only).
features should one consider as relevant that make an environment or object distinguishable? Depending on the context, one can take into account color, texture, Fourier or wavelet coefficients, etc. With robot vision, it is desirable to reduce the feature vector to classify objects in real time. For our robot navigation tasks, we have chosen only the object’s lines and planes as basic geometric features (see Fig. 8). Even though this yields a smaller number of features, it has proven to be very helpful for the description of complex 3D objects of offices and industrial environments. By coding these structures using planes, the robot can, at another time, recognize the environment or object by applying a classifier which is fed with a feature vector built with features taken from the 3D parameter Hough space.

6.2. Varying only the object angle

Let $O$ be a 3D object, represented by only planes ($n$ numbers of planes), and $O^{HT} = (\rho_1, \theta_1, \rho_2, \theta_2, \ldots, \rho_n, \theta_n)^T$ a Hough vector of these planes. The Hough representation is taken from the robot initial position. If the mobile robot moves, we get a new $O^{HT}$. Taking several samples and analyzing the vector, we can see that, only the value of $\rho$ and $\theta$ are changed. So, we can propose a new object feature representation; taking the planes from $O$, and getting its centroid $(x_c, y_c, z_c)$, then, we can compute the Hough Transform of the $O$ planes using its centroid as reference and store in $O_c^{HT}$. When the robot moves and gets a new $O_c^{HT}$, the values of $\phi_i \in O_c^{HT}$, $i = 1 \ldots n$, have a displacement in $[0, 2\pi)$. We can use (26) to get $\Delta \phi$, with the particularity that this value is equal for all $\phi_i$. To classify different objects, we can sort $O_c^{HT}$ by the $\rho$ value of every plane. Then apply a search for every object comparing their $\rho$ and $\theta$ values, and finding the displacements $\Delta \phi$ of them.

In Figs. 8–10 depicts tree simulated object (student workplace, stairway and indoor hall entry). The first row is the initial position where the object was taken. Second and third row shows the object taken from different places. First column is the position of the robot with respect to the object. Second column depict the Hough representation of the object taken from the robot position and shows the displacement (by a line) of the new representation as pink color. The third column depicts the Hough representation taken from the its centroid and shows the displacement (by a line) of the new representation as cyan color. As we can see in the third column, the value of $\phi$ is affected only by linear displacement, and consequently it is easy to classify these objects or environments because they have just 1 DOF.

7. Implementation and experimental results

In our next experiments, you will be able to appreciate the beauty and descriptive power of a set of planes stored in the Hough parameter space. In Fig. 11a we can see the robot looking in an environment and detecting with stereo vision the planes of the
students places. The robot is passing through this corridor that has many planes along its way. These planes are represented as points in the 3D Hough space, (see Fig. 11d) and processed by our algorithm. After the robot sensed the planes, Fig. 11b depicts them in

**Fig. 11.** (a) (From left first column) Robot view. (b) (Second column) Stereo images and virtual world. (c) (Third column) Planes represented in $\mathbb{R}^3$. (d) (four column) Planes in 3D parameter space.

**Fig. 12.** (a) (Top row) Robot perception of environments. (b) (Middle row) Planes represented in $\mathbb{R}^3$. (c) (Bottom row) Planes in 3D parameter space.
7.1. Perception of 3D shapes

Complex structures: in Fig. 12, the robot observes and codes using planes three different 3D structures. The first two are chosen to illustrate the role of planes to describe pure geometric structures. The third is a sort of industrial structure which is quite complex due its profusion of details. However, taking into consideration the more relevant planes, we are able to economically represent this complex structure. One can see the interesting result in the parameter Hough space, see Fig. 12c. The clusters of all three structures are remarkably different. Again, this facilitates the task of recognition using a classifier. One imagines a robot exploring environments where complex shapes are present; our approach can be of great use for efficient coding and recognition.

Sampling 3D objects with lines and planes is a data compression approach rather than using pointwise sampling. In this sense, our approach is inspired by the use of wavelets in 1D and 2D image processing.

8. Conclusions

In this paper, the authors have introduced the integration of Hough Transform to conformal geometric algebra. The 2D and 3D Hough Transforms are used as geometric descriptors of lines and planes, sensed by a laser range-finder and a stereo vision system for 3D map building and robot relocation. Once lines or planes are stored in these geometric sensors, one can compute relations or find geometric constraints useful for the detection of line and plane configurations, 3D map building and robot relocation, navigation, and obstacle avoidance. Also a new and simple 3D object descriptor was introduced. We believe that our approach can be of great use for mobile robots or robot humanoids.

Acknowledgments

The authors gratefully acknowledge the support of CINVESTAV and the project CONACYT-2007-1 No. 82084, “Cognitive and geometric methods for humanoid perception, learning, control and action.”

References