The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Generalized Hough Transform and Conformal Geometric Algebra to Detect Lines and Planes for Building 3D Maps and Robot Navigation

E. Bayro-Corrochano and M. Bernal-Marin

Abstract— **This paper describes a new approach for building 3D geometric maps using a geometric line and plane detector together with laser range finder and a stereo camera system. This detector is developed in such a way that one can related easily it with the conformal geometric algebra framework, thus detected lines and planes can be used for algebra of incidence computation to find geometric constraints useful to perceive special configurations in the 3D visual space for exploration, navigation and obstacle avoidance.**

I. Introduction

Mobile robots are equipped with different input devices to sense the surrounding environment. The laser range finder is widely used for this task due to its precision to detect lines, 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 with the shapes like walls and surrounding objects obtained by the sensors. This paper is an extension of previous work [1].

Using the Conformal Geometric Algebra 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: the occupancy grid map and geometric primitives based map. This work uses the latter. To know the pose (position an orientation) of the robot while the map is being building is a crucial need for mobile robots. In this work, for the robot relocalization, we use the line's characteristics in the Hough domain [2] (θ,ρ) for finding out 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 relocalize it in the same map In this work, we use the lines captured in the Hough domain to perform the relocalization inside the 2D map. Once the robot is relocalized, it uses the 3D Hough transform to detect planes and include them in the 3D map. Our robot algorithm exploits the 2D and 3D Hough space to relocalize and to find out particular 3D space configurations which can be used to recognized spacial areas and navigate safely. We present experiments using real data which validate the efficiency of our approach.

In section II we give an outline of the geometric algebra and the conformal geometric algebra. The 2D and the generalized Hough Transformation are presented in section three. Section four describes the approach for 3D map building. The procedure for the robot relocalization is explained in section five. The approach for the plane detection using the 3D Hough transform and stereo vision is presented in section six. The section seven is devoted to the concluding remarks.

E. Bayro-Corrochano and M.Bernal-Marin are with CINVESTAV, Unidad Guadalajara, Dept. Electrical Engineering and Conputer Science, Guadalajara, Mexico. edb,mbernal@gdl.cinvestav.mx

II. Geometric Algebra an Outline

The Geometric algebra $\mathcal{G}_{p,q,r}$ is constructed over the vector space $\mathcal{V}^{p,q,r}$, 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 $\mathcal{G}_{n=p+q+r}$ is 2^n, and $\mathcal{G}_n$ is constructed by the applications of the *geometric product* over the vector basis $\mathsf{e_i}$. The geometric product between two vectors $\mathbf{a}$,$\mathbf{b}$ is defined as

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}$$

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

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

$$\mathsf{e_i e_j} := \begin{cases} 1 \in \mathbb{R} & \text{for } i = j \in \{1, \ldots, p\} \\ -1 \in \mathbb{R} & \text{for } i = j \in \{p+1, \ldots, p+q\} \\ 0 \in \mathbb{R} & \text{for } i = j \in \{p+q+1, \ldots, n\} \\ \mathsf{e_{ij}} = \mathsf{e_i} \wedge \mathsf{e_j} & \text{for } i \neq j. \end{cases}$$

this lead in a basis for $\mathcal{G}_n$ that contains elements of different grade called *blades* (e.g. scalars, vectors, bivectors, trivectors, etc.):

$$1, \{\mathsf{e_i}\}, \{\mathsf{e_i} \wedge \mathsf{e_j}\}, \{\mathsf{e_i} \wedge \mathsf{e_j} \wedge \mathsf{e_k}\}, \cdots, \mathsf{e_1} \wedge \mathsf{e_2} \wedge \cdots \wedge \mathsf{e_n}$$

which is called *basis blade*; where the elements of maximum grade is the pseudoscalar $\mathtt{I} = \mathsf{e_1} \wedge \mathsf{e_2} \wedge \ldots \wedge \mathsf{e_n}$. A linear combination of basis blades, all of the same grade k, is called k-vector. The linear combination of such k-vectors is called *multivector*, and multivectors witch certain characteristics represent different geometric objects or entities (as points, lines, planes, circles, spheres, etc.), depending on the GA where we are working (for example, a point (a, b, c) is represented in $\mathcal{G}_{3,0,0}$ [the GA of the 3D-Euclidean space $\mathcal{E}^3$] as $\mathbf{x} = a\mathsf{e_1} + b\mathsf{e_2} + c\mathsf{e_3}$, however a circle can not be defined in $\mathcal{G}_{3,0,0}$, but it is possible to define it in $\mathcal{G}_{4,1,0}$ (CGA) as a 4-vector $\underline{z} = \underline{s_1} \wedge \underline{s_2}$ [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 $< M >_r$ where r is the grade of the blades we

want to extract (obtaining an homogeneous multivector *M'* or a r-vector).

The *dual* $\mathbf{X}^*$ of a *r*-blade $\mathbf{X}$ is defined by $\mathbf{X}^* = \mathbf{X}\mathtt{I}_\mathtt{n}^{-1}$. It follow that the dual of a *r*-blade is an $(n-r)$-blade.

The *reverse* of any multivector *M* is defined as

$$\langle \widetilde{M} \rangle_i = (-1)^{\frac{i(i-1)}{2}} \langle M \rangle_i, \text{ for } M \in \mathcal{G}_n, 0 \leq i \leq n. \quad (1)$$

The reader should consult [3] to detailed explanation about CGA and its applications.

A. Conformal Geometric Algebra

To work 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 particular meaning; in this way we represent particular entities of the Euclidean space with subspaces of the conformal space. The

TABLE I
ENTITIES IN CGA

Entity	IPNS	OPNS
Sphere	$\underline{s} = \mathbf{p} + \frac{1}{2}(\mathbf{p}^2 - \rho^2)\mathtt{e} + \mathtt{e}_0$	$\underline{s}^* = \underline{a} \wedge \underline{b} \wedge \underline{c} \wedge \underline{d}$
Point	$\underline{x} = \mathbf{x} + \frac{1}{2}\mathbf{x}^2\mathtt{e} + \mathtt{e}_0$	$\underline{x}^* = (-\mathtt{E}\mathbf{x} - \frac{1}{2}\mathbf{x}^2\mathtt{e} + \mathtt{e}_0)\mathtt{I}_\mathtt{E}$
Plane	$\underline{P} = \mathbf{N}\mathtt{I}_\mathtt{E} - d\mathtt{e}$ $\mathbf{N} = (\mathbf{a}-\mathbf{b}) \wedge (\mathbf{a}-\mathbf{c})$ $d = (\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c})\mathtt{I}_\mathtt{E}$	$\underline{P}^* = \mathtt{e} \wedge \underline{a} \wedge \underline{b} \wedge \underline{c}$
Line	$\underline{L} = \underline{P_1} \wedge \underline{P_2}$ $= \mathbf{r}\mathtt{I}_\mathtt{E} + \mathtt{e}\mathbf{M}\mathtt{I}_\mathtt{E}$ $\mathbf{r} = \mathbf{a} - \mathbf{b}$ $\mathbf{M} = \mathbf{a} \wedge \mathbf{b}$	$\underline{L}^* = \mathtt{e} \wedge \underline{a} \wedge \underline{b}$
Circle	$\underline{z} = \underline{s_1} \wedge \underline{s_2}$ $\underline{s_z} = (\mathtt{e} \cdot \underline{z})\underline{z}$ $\rho_z = \frac{\underline{z}^2}{(\mathtt{e} \wedge \underline{z})^2}$	$\underline{z}^* = \underline{a} \wedge \underline{b} \wedge \underline{c}$
P-pair	$\underline{PP} = \underline{s_1} \wedge \underline{s_2} \wedge \underline{s_3}$	$\underline{PP}^* = \underline{a} \wedge \underline{b}$

extra vectors we add are $\mathtt{e}_+$ and $\mathtt{e}_-$, defined by the properties ${\mathtt{e}_+}^2 = 1, {\mathtt{e}_-}^2 = -1, \mathtt{e}_+ \cdot \mathtt{e}_- = 0$. With this two vectors, we define the null vectors

$$\mathtt{e}_0 = \frac{1}{2}(\mathtt{e}_- - \mathtt{e}_+); \qquad \mathtt{e} = \mathtt{e}_- + \mathtt{e}_+ \quad (2)$$

interpreted as the origin and the point at infinity, respectively. From now on and in the rest of the paper, points in the 3D-Euclidean space are represented in lowercase, while conformal points in underline letters; also the conformal entities will be expressed in the *Outer Product Null Space* (OPNS) (noted with an asterisk beside, also know as the dual of the entity), and no in the *Inner Product Null Space* (IPNS) (without 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 $\mathbf{x} \in \mathcal{V}^3$ to the Conformal space in $\mathcal{G}_{4,1}$ (using IPNS) we use

$$\underline{x} = \mathbf{x} + \frac{1}{2}\mathbf{x}^2\mathtt{e} + \mathtt{e}_0 \quad (3)$$

Applying the wedge operator "$\wedge$" on points, we can express new entities in CGA. All geometric entities from CGA are show in the table I for a quick reference.

The pseudoscalar in CGA $\mathcal{G}_{4,1,0}$ is defined as

$$\mathtt{I} = \mathtt{I}_\mathtt{E}\mathtt{E} \quad (4)$$

where $\mathtt{I}_\mathtt{E} = \mathtt{e}_1\mathtt{e}_2\mathtt{e}_3$ is the pseudoscalar from $\mathcal{G}_3$ and $\mathtt{E} = \mathtt{e}_+\mathtt{e}_-$ is the pseudoscalar from the Minkowski plane.

In GA there exist specific operators to model rotations and translations called *rotors* and *translators* respectively. In CGA such operator are called *versor* and are defined by (5) being $\mathbf{R}$ the rotor, $\mathbf{T}$ the translator.

$$\mathbf{R} = e^{-\frac{1}{2}\underline{l}\theta}; \qquad \mathbf{T} = e^{\frac{\mathtt{e}\mathbf{t}}{2}}, \quad (5)$$

where the *rotation axis* $\underline{l} = l_1\mathtt{e}_{23} + l_2\mathtt{e}_{31} + l_3\mathtt{e}_{12}$ is a unit bivector which represents a line (in IPNS) through the origin in CGA, θ is the rotation angle, $\mathbf{t} = t_1\mathtt{e}_1 + t_2\mathtt{e}_2 + t_3\mathtt{e}_3$ is the translation vector in $\mathcal{V}^3$. The equations (5) can also be expressed as

$$\mathbf{R} = cos\left(\frac{\theta}{2}\right) - sen\left(\frac{\theta}{2}\right)\underline{l}; \qquad \mathbf{T} = (1 + \frac{\mathtt{e}\mathbf{t}}{2}) \quad (6)$$

due to the exponential properties. Such operator are applied to any entity of any dimension by multiplying the entity by the operator from the left, and by the *reverse* of the operator from the right, as show in (7).

$$\underline{x}' = \sigma\underline{x}\widetilde{\sigma} \quad (7)$$

where $\underline{x}$ is any entities mentioned in table I, and σ is a versor (*rotor*, *translator* or *motor* mentioned below). Using (7) is easily to transform any entities from CGA (points, point-pair, lines, circles, planes, spheres), not only points as is usual in other algebras.

In CGA it is possible to use the rotors and translator to express general rotation and screw motions in space. To model a screw motion, the entity has to be translated during a general rotation with respect to the rotation axis. The implementation consecutive of a translator and rotor can be written as the product of them. Such operator is called *motor* and expressed as

$$\mathbf{M} = \mathbf{TR} \quad (8)$$

The translator, rotor and motor (all of them *versors*) are elements from $\mathcal{G}^+_{4,1}$, and they defines an algebra called *motor algebra*. This algebra greatly simplifies the successive computation of rotations and translation, applying only the geometric product in consecutive versors, giving the final result another versor of this algebra, where all the transformations are together in one element.

III. 2D AND 3D HOUGH TRANSFORM

The Hough Transform [2] is a robust and effective method to identify the location and orientation of lines. The transform is the parametrization of a line from the (x,y) plan (a Cartesian plan) to the (θ,ρ) plan (the Hough domain).

A. 2D Hough Transform

The line segments of the map are transformed to the Hough domain, defining the transformation in the domain of $\theta \in [0, 2\pi)$, so every line segment in (x,y) correspond to a point (θ,ρ). This gives us one characteristic in a line, if it

varies only in its angle θ it keeps the value of ρ constant. So given a previous captured map $\mathcal{G}$ (global map) and a new captured map $\mathcal{L}$ (new local map) the difference between them is an angle $\Delta\theta$ and a displacement Δx and Δy which affects the ρ value.

The difference of an angle in the Hough domain is defined as follow

$$\Delta\theta(\theta_a, \theta_b) = \begin{cases} \theta_a - \theta_b - 2\pi & \text{if case 1} \\ \theta_a - \theta_b + 2\pi & \text{if case 2} \\ \theta_a - \theta_b & \text{if other} \end{cases} \quad (9)$$

where

case 1 : if $\theta_a > \theta_b$ and $\theta_a + \theta_\xi \geq 2\pi$ and $\theta_b + \theta_\xi \leq 0$
case 2 : if $\theta_a < \theta_b$ y $\theta_b + \theta_\xi \geq 2\pi$ y $\theta_a + \theta_\xi \leq 0$

this give us the calculus of an any point near by other where its angles are near to $0 = 2\pi$.

B. Generalized Hough Transform

The 2D hough transform maps the $\mathbb{R}^2$ to a the so called 2D Hough space parametrized by (θ, ρ). Since in this work, we represent geometric entities in conformal geometric algebra, we need to reformulate the Hough transform so that we can use comfortably to 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 possibly to generalize it for any hyperplane of $\mathbb{R}^n$, where $n = 3, 4, \ldots$. However, the way we represent such hyperplanes is the key, thus we resort to formulate a generalized Hough transform related with the conformal geometric algebra. The intrinsic characteristics of Hyperplanes are just two: Hesse distance (a scalar) and hyperplane orientation (nD unit vector), thus a hyperplane in $\mathbb{R}^n$ has n+1 D.O.F. What we will now formulate, it will be called from know on the Hough geometric sensor. First of all, we should underline the fact that any Hough space is just a parametrized space, which can be put in connection with conformal geometric algebra for the purpose of carrying out algebraic computation using the hyperplanes coded in the Hough space in question. That means that a nD Hough space will be used together with a $\mathcal{G}_{n+1,1}$ conformal geometric algebra. Our formalism was inspired by the concept of duality between hyperplanes and points both represented in homogeneous coordinates. Thus, if we imagine in $\mathbb{R}^n$ centered hyperspheres of varying radius (perfect set), we can consider the radius as Hesse distances an any point on the surface of any hypersphere as an unique hyperplane, see figure 1. Now, any point lying on this hyperplane $\mathbf{p} \wedge \pi = 0$ increments a counter of this plane. This model is straightforward extension of the standard 2D Hough transform. Each counter can store also (n-1) coefficients linked with each point solely to identify the point with respect to the plane. Finally, we represent this points in terms of two polar coordinates and the Hesse distance. Each counter lying on the surface will represent the intersection of infinity undulated manifolds, i.e. in 2D are sinusoidal and in 3D surfaces with sinus undulation.

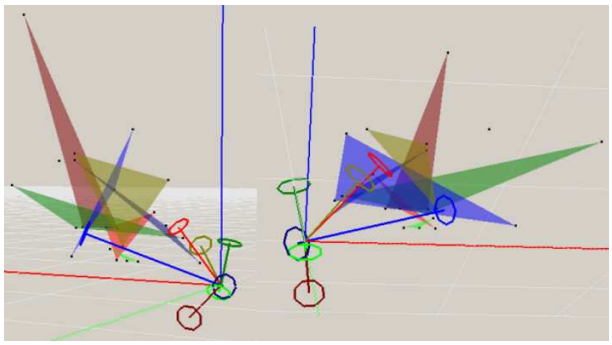

Fig. 1. Some planes and its Hesse normal form representation

The geometric sensor can be used by a robot vision system to detect planes in the visual 3D space. Once stable planes are found using robust algorihms like RANSAC [4][5], we can use the parameters of the 3D plane (Hesse distance and plane orientation) to represent a plane in 3D conformal geometric algebra $\mathcal{G}_{n+1,1}$ and look for geometric constrains between points, lines and planes. For that we make use of algebra of incidence using teh meet and join operations. The discovery of geometric constrains is of extrem use for helping the robot to navigate in places with obstacles. See in section VI an interesting formulation of this new formulation.

IV. 3D Map Building

Using an equipped mobile robot with a laser rangefinder sensor and stereo camera system mounted on a pan-tilt head, each one with their own coordinate system. We apply *the method of hand-eye calibration* [6] to get the center coordinates of each devices related to a global robot coordinate system. While the robot is moving exploring the new areas two maps are performed simultaneously, one with local coordinates "$\mathcal{L}$" (according to the current reading of the robot) and the other with global coordinates "$\mathcal{G}$" (according to the initial position of exploration). The use of the encoders help us to estimate the actual position of the mobile robot but this lectures has errors due to frictions on the wheels. Therefore the pose of the robot, its rotation angle and translation are calculated by

$$\theta = \theta_o + \theta_{error} \quad (10)$$
$$T = T_o + T_{error} \quad (11)$$

where θ_o and T_o are the rotation angle and the translation vector given by the odometer, and θ_{error} and T_{error} are the value of 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 the perpendicular line to plane (x, y) as rotation axis and (10), and adding a third fixed coordinate to (11) we can apply this values in (5) to make $\mathbf{T_{pos}}$ and $\mathbf{R_{pos}}$ that represent the movement of the robot in the environment.

In the next section we explain how to model data from the input devices in the 3D environment.

A. Line detection

To extract line segments from range points, we use recursive line splitting method as show in [7], this is a speedy and correctness algorithm that performs *divide-and-conquer* algorithms. For every endpoints of the line segments, we maps them to CGA to get the pair of points entity and store in a local map $\mathcal{L}$. As the endpoints are 2D points we take the last coordinate in $\mathcal{V}^3$ and give the 0 value to fix the point in that plane. Now the local map $\mathcal{L}$ has every line segments represented as pair of points in CGA and we can apply any transformation on it (rotation, translation). While the map is being built the collected data is stored in it with regard the initial position The following records taken from the laser rangefinder replace the actual local map for every new robot position in the environment. When a new local map $\mathcal{L}$ is taken, it is mapped to the global coordinate system using (15) to perform a line matching. Here we use one property of sphere to matching line segments (pair of point), namely having two spheres $\underline{s_1}^*$ and $\underline{s_1}^*$ the product

$$(\underline{s_1} \wedge \underline{s_2})^2 \begin{cases} < 0 & \text{if } \underline{s_1} \text{ and } \underline{s_2} \text{ intersect} \\ = 0 & \text{if } \underline{s_1} \text{ and } \underline{s_2} \text{ are tangent} \\ > 0 & \text{if } \underline{s_1} \text{ and } \underline{s_2} \text{ don't intersect,} \end{cases}$$

and to get a sphere from pair of points we use

$$\underline{S_{PP^*}} = \frac{\underline{PP^*}}{\underline{PP^*} \wedge \mathsf{e}} \tag{12}$$

When we got the line matching, we merge both maps and correct the angle and displacement of the lines comparing between local and global map, this little error is caused by the odometry sensor. Then update the actual position of the robot using (10) and (11).

We can express a motor that maps any entity that have been taken from the laser coordinates system to the global coordinates system. Taking the laser's center of coordinates, and making a motor $\mathbf{M_{lsr}}$ that represent the rotation and translation from the center of the global coordinates system to the laser's center, and developing

$$\mathbf{M_{cl}} = \mathbf{R_{pos}}\mathbf{M_{lsr}}\widetilde{\mathbf{R_{pos}}} \tag{13}$$

$$\mathbf{M_{pos}} = \mathbf{T_{pos}}\mathbf{R_{lsr}} \tag{14}$$

$$\mathbf{M_{lu}} = \mathbf{M_{cl}}\mathbf{M_{pos}} \tag{15}$$

where (13) is the translation and rotation motor toward laser's center; (14) is the movement of robot using laser rangefinder and (15) is the motor which leads us to the source of the laser sensor in the global coordinate system.

Using (15) with any geometric entity (points, lines, circles) recorded with the laser rangefinder sensor, we can move easily to the global coordinate system using the form

$$\underline{x}' = \mathbf{M_{lu}}\underline{x}\widetilde{\mathbf{M_{lu}}} \tag{16}$$

As we are dealing in a 3D real world and the laser rangefinder only show us a plane measure, we can add a virtual wall (fig. 2) to the shapes from laser rangefinder to get a 3D visual sense of the walls that are inside of the virtual world.

If a new laser rangefinder is mounted on the mobile robot or if the laser rangefinder is moved in another place in the mobile robot, is easy to get the new motor that maps the data from laser rangefinder to the global map, only updating the motor $\mathbf{M_{lsr}}$ that represent the rotation and translation from the center of the global coordinates system to the laser's center, and recalculate (15).

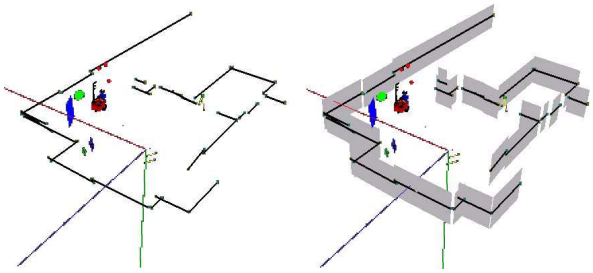

Fig. 2. 3D map using virtual walls

B. Stereo camera system with pan-tilt unit

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 show in (5). Let $\mathbf{R_{pan}}$ be the rotor for the pan movement and let $\mathbf{R_{tilt}}$ for the tilt movement. Applying this rotors using the geometric product we can model all the pant-tilt system. The stereo camera system has is center coordinates on the left camera (right camera viewing in front). We apply *the method of hand-eye calibration* [6], to get the axis from the pan-tilt unit and getting its intersection (or the closet point between the rotation axis), we build a translation from this intersection to the source of the stereo camera system. This translation is performed using a translator $\mathbf{T_{eye}}$ as show in (5). Now getting all this information we develop a motor that maps any entities taken from the stereo camera system to the global coordinates system as

$$\mathbf{T_{ap}} = \mathbf{R_{pos}}\mathbf{T_{axis}}\widetilde{\mathbf{R_{pos}}} \tag{17}$$

$$\mathbf{R_{pt}} = \mathbf{R_{pos}}\mathbf{R_{pan}}\mathbf{R_{tilt}} \tag{18}$$

$$\mathbf{T_{opt}} = \mathbf{R_{pt}}\mathbf{T_{eye}}\widetilde{\mathbf{R_{pt}}} \tag{19}$$

$$\mathbf{M_{mpt}} = \mathbf{T_{pos}}\mathbf{R_{pt}} \tag{20}$$

$$\mathbf{M_{su}} = \mathbf{T_{opt}}\mathbf{T_{ap}}\mathbf{R_{mpt}} \tag{21}$$

where (17) 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. (18) is the rotor resulting of all the spins that has done so much in the position of the robot, as in the pan-tilt. (19) is the translation to the left camera of the stereo camera system taking into account all the movements that had the system. (20) is the movements motor of the robot, along with the pan-tilt. (21) is the complete movement motor of the robot.

Any point captured by the cameras in any angle of the pan-tilt unit, in any position of the robot can be map from

the stereo camera system to global coordinate system using the form

$$\underline{x}' = \mathbf{M_{su}}\underline{x}\widetilde{\mathbf{M_{su}}} \tag{22}$$

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 (22).

C. Plane detection

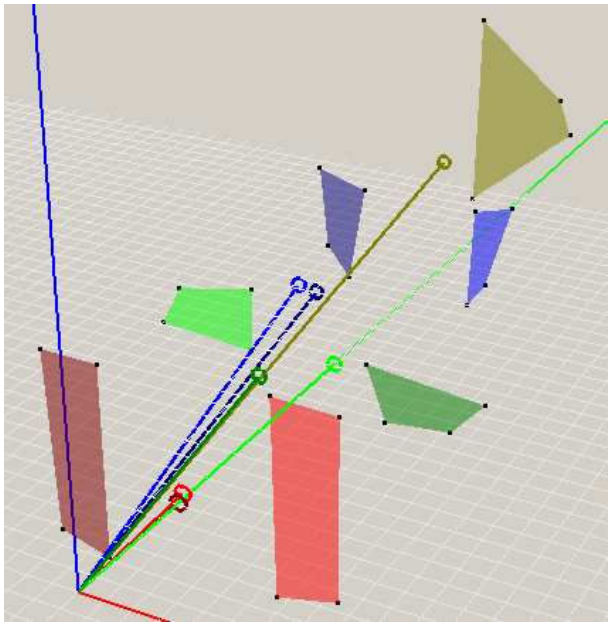

Fig. 3. Planes in a corridor and their Hesse form representation

Points obtained from stereo images and formulated using equation (22) can be used to obtain a representation of a plane in virtual world. Several techniques can be used to get features from images. 3D plane parameters can obtained using stereo vision and the RANSAC. The plane parameters are the radius or radial distance ρ, the plane tilt (or polar angle) θ and the azimuth (or azimuthal angle) φ (see Fig. 3). Using these parameters, we get the Hesse distance of the plane. The plane representation in the environment is displayed using the points lying on the plane. Four of these points are used to represent the plane.

The ρ, θ, φ are utilized in the Hough domain and are used as point in this space. Using these points is easier to handle planes due to its values ($\rho > 0$, $\theta \in [0, \pi)$ and $\varphi \in (-\pi, \pi]$). When the mobile robots navigate, only the value of ρ and θ are changed. Note, that A plane is represented just by 3 DoF.

V. Robot Relocalization on a Map

According subsection III-B, the Hough transform is the parametrization of a line from the (x,y) plan (a Cartesian plan) to the (θ,ρ) plan (the Hough domain). The line segments of the map are transformed to the Hough domain, defining the transformation in the domain of $\theta \in [0, 2\pi)$, so every line segment in (x,y) correspond to a point (θ,ρ). This gives us one characteristic in a line, if it varies only in its angle θ it keeps the value of ρ constant. So given a previous captured map $\mathcal{G}$ (global map) and a new captured map $\mathcal{L}$ (new local map) the difference between them is an angle $\Delta\theta$ and a displacement Δx and Δy which affects the ρ value.

The difference of an angle in the Hough domain is defined by equation 9. The relocation follows the next steps:

- Extract the actual environment, using the laser rangefinder, extract the line segment and map them to the Hough domain and store in $\mathcal{L}$ ($\mathcal{L}$ only has (θ,ρ) from each line).
- Make the difference for each element in $\mathcal{L}$ with each element in $\mathcal{G}$ (using (9) in angles) and store it in $\Delta_{(\theta,\rho)}$, giving us a twist and displacement, this step can see as the difference of the actual map an the previous captured.

$$\Delta_{(\theta,\rho)} = \mathcal{G} - \mathcal{L} \tag{23}$$

- Now we build a new global map adding all the elements of $\Delta_{(\theta,\rho)}$ to an element $l_i \in L$ and store it in $\mathcal{G}'_i$ as show in (24)

$$\mathcal{G}' = \Delta_{(\theta,\rho)} + \mathcal{L}_i \tag{24}$$

  this give us a displacement of the actual map close to the global map.
- Now the angle $\Delta\theta$ have been shifted in $\mathcal{G}'_i$, so we decrease $\mathcal{G}$ by the value of $\mathcal{G}'_i$, and get an error of displacement ξ_i. The goal is to reduce this error using

$$\sum \mathcal{G}'_i - \mathcal{L} = 0 \tag{25}$$

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

$$\xi_i < (\xi_\theta, \xi_\rho) \tag{26}$$

  where ξ_θ and ξ_ρ are threshold from the angle and the ρ respectively.

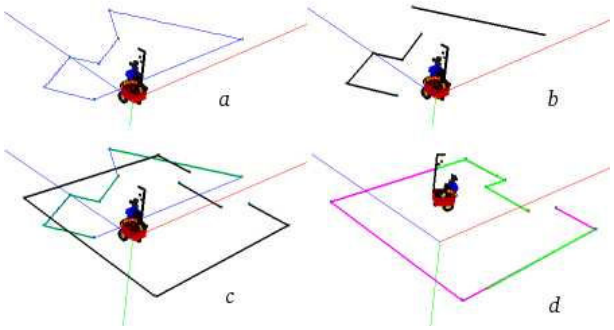

Fig. 4. Steps in relocation

Repeat last 3 steps for each line in $\mathcal{L}$. Finally when all the line where displaced and voted and extracting the maximum value per column from V where the row position correspond to a line in $\mathcal{G}$ so this is the line correspondence, if the value is null, there is not 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 according to the samples taken.

Using each matching line to get the average of the angles and with this angle get the rotation angle to build a new *rotor* to turn the robot and the local map $\mathcal{L}$ (line segments) in the new environment. Now we have the orientation of the mobile robot and is only missing the displacement position. We can find the displacement Δx and Δy using the closet point to the origin in the matching lines, to generate a translation vector. The closet point in to the origin on a line in CGA can be calculate by

$$\mathbf{p} = -(\underline{L}^* \cdot \mathtt{E}) \cdot ((\mathtt{e}_+ \cdot \underline{L})\mathtt{I}_\mathtt{E}) \tag{27}$$

as we get line segments (pair of points in CGA) only need to apply the wedge operator with the point at infinity as show in (28)

$$\underline{L}^* = \underline{PP}^* \wedge \mathtt{e} \tag{28}$$

to get the line in CGA and perform (27). With the translation vector we make a *translator* and apply it to the local map and to the mobile robot. And now the robot is locate in the correct place into the map, then we can continue with navigation within the environment. In Fig. 4 we can see the relocation evolution, where (a) show the initial position of the robot and it is taking a sample of the environment, (b) generating the lines segments of the actual environment (c) load previous map to perform matching, here we can see that the mobile robot is displaced and turned in an random place, (d) locate and put the robot in the correct place into the map, here the robot is located itself on the previous environment and placed in the right place.

VI. Plane Detection for a 3D Map

Figure. 5 shows a mobile robot exploring a room. The robot is passing throw a corridor, that has many planes on its way. These planes are sensed by stereo camera system. Planes in front of the robot are detected and processed by our algorithm. At the bottom left of the figure, we see the left image of the stereo vision system, where the planes were highlighted with colour. On the right, after the robot sensed the planes, it depicts them in a virtual environment. As you can see, the planes are positioned according to their right spatial location which was observed by the stereo vision system. All these planes have their representation in CGA and in the 3D Hough space. Note, that We can pass a plane represented in CGA to a plane representation in the 3D Hough space.

VII. Conclusions

In this paper the authors have introduced the generalized Hough transform related with conformal geometric algebra. The 2D and 3D Hough transforms are used as geometric detectors of lines and planes sensed by a laser range finder and a stereo vision system for 3D map building and robot relocalization . Once lines or planes are stored in theses geometric sensors, one can compute relations or find geometric constrains useful for line and plane configurations detection, 3D map building and robot relocalization, navigation an obstacle avoidance. We believe that our approach can be of great use for mobile robots or robot humanoids.

Fig. 5. A mobile robot in a corridor with planes by its side

VIII. 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

[1] M. Bernal-Marin and E. Bayro-Corrochano, “Visual and laser guided robot relocalization using lines, hough transformation and machine learning techniques,” in *IROS*. IEEE, 2009, pp. 4142–4147.

[2] P. Hough, “Methods and means for recognizing complex patterns,” *U.S. Patent 3,069,654*, 1962.

[3] E. Bayro-Corrochano, *Robot perception and action using conformal geometry*, 1st ed. Springer Verlag, 2005, ch. 13, pp. 405–458.

[4] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[5] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer, “Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data,” in *ISPRS Workshop on Laser Scanning*, 2007, p. 407.

[6] E. Bayro-Corrochano, K. Daniilidis, and G. Sommer, “Motor algebra for 3d kinematics: The case of the hand-eye calibration,” *Journal of Mathematical Imaging and Vision*, vol. 13, no. 2, pp. 79–100, 2000.

[7] L. Zhang and B. K. Ghosh, “Line segment based map building and localization using 2d laser rangefinder,” in *ICRA*. IEEE, 2000, pp. 2538–2543.