**The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA**

Visual and Laser Guided Robot Relocalization Using Lines, Hough Transformation and Machine Learning Techniques

Miguel Bernal-Marin and Eduardo Bayro-Corrochano

Abstract— **This paper describes a new approach for building 3D geometric maps using a laser rangefinder, a stereo camera system and a mathematical system the Conformal Geometric Algebra. The use of a known visual landmarks in the map helps to carry out a good localization of the robot. These landmarks are found using the Viola and Jones algorithm and are represented with their position in the 3D virtual map. This landmarks help in the relocalization of a robot in a previously captured environment. This machine learning technique is used for recognition of objects in the environment.**

I. Introduction

Mobile robots are equipped with multiple input devices to sense the surrounding environment. The laser rangefinder is widely used for this task due to its precision, and its wide capture range. In this paper we merged the data obtained by the laser and the stereo camera system to build a 3D virtual map with the shapes obtained by these devices. The 3D objects seen by the stereo camera system can be modeled by geometric entities, which are easy to represent and to combine. Some of these 3D objects can act as a landmarks for the robot navigation and relocalization. Line segments are used to build a 3D map and they are the most widely used features [1] [2].

Using the Conformal Geometric Algebra we can represent different geometric shapes including the line segments (as a pair of points) and the data captured by the stereo camera system (landmarks as labeled spheres). This framework also allows us to formulate transformations (rotation, translation) using spinors or versors.

For relocalization we use the line's characteristics in the Hough domain [3] (θ,ρ), for find out the current robot position in the 2D map.

We present experiments using real data which validate the efficiency of our approach.

II. Geometric Algebra

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 e_i. The geometric product between two vectors $\mathbf{a}$,$\mathbf{b}$ is defined as

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

E. Bayro-Corrochano and M.Bernal-Marin are with Center for Research and Advanced Studies of the National Polytechnic Institute, Guadalajara, Mexico `edb,mbernal@gdl.cinvestav.mx`

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

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

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 [4] 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 extra vectors we add are e_+ and e_-, defined by the properties

${e_+}^2 = 1, {e_-}^2 = -1, e_+ \cdot e_- = 0$. With this two vectors, we define the null vectors

$$e_0 = \frac{1}{2}(e_- - e_+); \qquad e = e_- + e_+ \tag{2}$$

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

$$\underline{x} = \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e + e_0 \tag{3}$$

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

$$\mathtt{I} = \mathtt{I_E E} \tag{4}$$

where $\mathtt{I_E} = e_1 e_2 e_3$ is the pseudoscalar from $\mathcal{G}_3$ and $\mathtt{E} = e_+ e_-$ is the pseudoscalar from the Minkowski plane.

TABLE I
ENTITIES IN CGA

Entity	IPNS	OPNS
Sphere	$\underline{s} = \mathbf{p} + \frac{1}{2}(\mathbf{p}^2 - \rho^2)e + e_0$	$\underline{s}^* = \underline{a} \wedge \underline{b} \wedge \underline{c} \wedge \underline{d}$
Point	$\underline{x} = \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e + e_0$	$\underline{x}^* = (-\mathtt{E}\mathbf{x} - \frac{1}{2}\mathbf{x}^2 e + e_0)\mathtt{I_E}$
Plane	$\underline{P} = \mathbf{N}\mathtt{I_E} - de$	$\underline{P}^* = e \wedge \underline{a} \wedge \underline{b} \wedge \underline{c}$
	$\mathbf{N} = (\mathbf{a} - \mathbf{b}) \wedge (\mathbf{a} - \mathbf{c})$	
	$d = (\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c})\mathtt{I_E}$	
Line	$\underline{L} = \underline{P_1} \wedge \underline{P_2}$	$\underline{L}^* = e \wedge \underline{a} \wedge \underline{b}$
	$= \mathbf{r}\mathtt{I_E} + e\mathbf{M}\mathtt{I_E}$	
	$\mathbf{r} = \mathbf{a} - \mathbf{b}$	
	$\mathbf{M} = \mathbf{a} \wedge \mathbf{b}$	
Circle	$\underline{z} = \underline{s_1} \wedge \underline{s_2}$	$\underline{z}^* = \underline{a} \wedge \underline{b} \wedge \underline{c}$
	$\underline{s_{\underline{z}}} = (e \cdot \underline{z})\underline{z}$	
	$\rho_{\underline{z}} = \frac{\underline{z}^2}{(e \wedge \underline{z})^2}$	
P-pair	$\underline{PP} = \underline{s_1} \wedge \underline{s_2} \wedge \underline{s_3}$	$\underline{PP}^* = \underline{a} \wedge \underline{b}$

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.

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

where the *rotation axis* $\underline{l} = l_1 e_{23} + l_2 e_{31} + l_3 e_{12}$ is a unit bivector which represents a line (in IPNS) through the origin in CGA, θ is the rotation angle, $\mathbf{t} = t_1 e_1 + t_2 e_2 + t_3 e_3$ is the translation vector in $\mathcal{V}^3$. The equations (5) can also be expressed as

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

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).

$$\underline{x'} = \sigma \underline{x} \widetilde{\sigma} \tag{7}$$

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

$$\mathbf{M} = \mathbf{TR} \tag{8}$$

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.

Vector calculus is a coordinate dependent mathematical system and its cross product can not be extended to higher dimensions. The representation of geometric primitives is based in lengthy 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.

III. 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* [5] 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

$$\theta = \theta_o + \theta_{error} \tag{9}$$
$$T = T_o + T_{error} \tag{10}$$

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 (9), and adding a third fixed coordinate to (10) 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. Laser rangefinder

To extract line segments from range points, we use recursive line splitting method as show in [1] [2], this is a speedy and correctness algorithm that performs *divide-and-conquer* algorithms [6]. For every endpoints of the line segments, we maps them to CGA to get the pair of points entity (see table I) 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 (14) 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

$$\underline{S_{PP^*}} = \frac{\underline{PP}^*}{\underline{PP}^* \wedge \mathsf{e}} \tag{11}$$

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 (9) and (10).

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

$$\begin{aligned} \mathbf{M_{cl}} &= \mathbf{R_{pos}} \mathbf{M_{lsr}} \widetilde{\mathbf{R_{pos}}} && (12) \\ \mathbf{M_{pos}} &= \mathbf{T_{pos}} \mathbf{R_{lsr}} && (13) \\ \mathbf{M_{lu}} &= \mathbf{M_{cl}} \mathbf{M_{pos}} && (14) \end{aligned}$$

where (12) is the translation and rotation motor toward laser's center; (13) is the movement of robot using laser rangefinder and (14) is the motor which leads us to the source of the laser sensor in the global coordinate system.

Using (14) 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

$$\underline{x}' = \mathbf{M_{lu}} \underline{x} \widetilde{\mathbf{M_{lu}}} \tag{15}$$

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. 1) 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 (14).

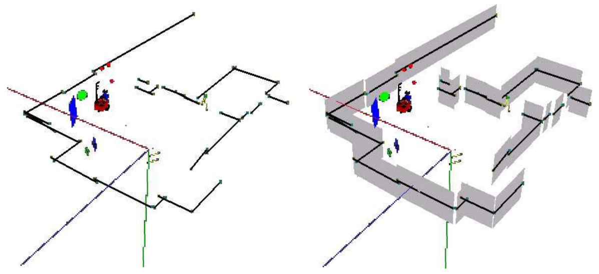

Fig. 1. 3D objects in the map and the virtual walls

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* [5], 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

$$\begin{aligned} \mathbf{T_{ap}} &= \mathbf{R_{pos}} \mathbf{T_{axis}} \widetilde{\mathbf{R_{pos}}} && (16) \\ \mathbf{R_{pt}} &= \mathbf{R_{pos}} \mathbf{R_{pan}} \mathbf{R_{tilt}} && (17) \\ \mathbf{T_{opt}} &= \mathbf{R_{pt}} \mathbf{T_{eye}} \widetilde{\mathbf{R_{pt}}} && (18) \\ \mathbf{M_{mpt}} &= \mathbf{T_{pos}} \mathbf{R_{pt}} && (19) \\ \mathbf{M_{su}} &= \mathbf{T_{opt}} \mathbf{T_{ap}} \mathbf{R_{mpt}} && (20) \end{aligned}$$

where (16) 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. (17) is the rotor resulting of all the spins that has done so much in the position of the robot, as in the pan-tilt. (18) is the translation to the left camera of the stereo camera system taking into account all the movements that had the system. (19) is the movements motor of the robot, along with the pan-tilt. (20) 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

$$\underline{x}' = \mathbf{M_{su}} \underline{x} \widetilde{\mathbf{M}_{\mathbf{su}}} \tag{21}$$

Using the CGA we can capture all the entities showed in the table I, 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 (21).

With (21) and (15) all entities (stereo camera and laser rangefinder data) can merge in the same 3D virtual environment. This combined data give us the location from the 3D object with respect to the mobile robot in the real environment.

IV. Relocation in a map

The location of a robot in an environment already captured is one of the problems that arise once the robot has finished the full map of its environment, and it is moved to an arbitrary place within it. The goal is to relocate the mobile robot within the map previously captured. This is known as the ”kidnapping problem”. The map that has been made, has geometric information of the surrounding environment. From such, we only use the records obtained by the laser rangefinder sensor, as lines are less noise sensitive.

The Hough Transform [3] 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). 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 two angles θ_a and θ_b in the Hough domain is defined as follow

$$\Delta\theta(\theta_a, \theta_b) = \begin{cases} \theta_a - \theta_b - 2\pi & \text{if case 1} \\ \theta_a - \theta_b + 2\pi & \text{if case 2} \\ \theta_a - \theta_b & \text{if other} \end{cases} \tag{22}$$

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 $0 = 2\pi$.

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). See fig. 2 (a) and (b).
- Make the difference for each element in $\mathcal{L}$ with each element in $\mathcal{G}$ (using (22) 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. See fig. 2 (c)

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

- 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 an the ρ respectively.

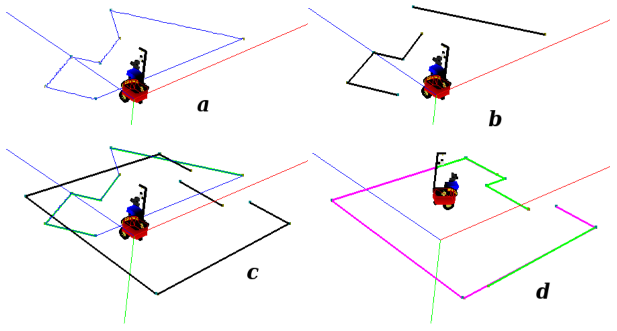

Fig. 2. Steps in relocation

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. 2 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.

Using these steps on a line based map, the mobile robot can get its position taking small samples of the surrounding environment. Only the line parameters (θ,ρ) are need to performs the re-localization.

V. Getting 3D positions based on visual landmarks

A landmark literally is a geographic feature used by explorers and others to find their way back or move through an area. In the map building process a mobile robot can use these landmarks to remember the place where it was before while it explore its environment. Also the landmarks can be used to find robot position in a previous building map facilitating the re-localization. As we are using a camera stereo system, the 3D position of any object can be also calculated and it can be represented in the 3D virtual environment. Using these objects as a landmarks, the robot gets its relative position.

A. Machine learning phase

A natural or artificial landmark located in the actual environment helps to the mobile robot to know its position on the map. Viola and Jones present a new and radically faster approach to face detection based on the AdaBoost algorithm from machine learning [7], and this approach can be used to detect our statics landmarks. Once the landmarks have been selected and trained, the mobile robot can use them to navigate in the environment performs the Viola an Jones algorithm. If a landmark is found we get a sub-image I_L from the left camera image. This I_L is the region of the image where the landmark was found (fig. 3).

When a landmark is identified in one image (left camera), we must be sure that the landmark is in the other image as well (right camera of the stereo camera system). To get the 3D position, the landmark must be detected in both images. The landmark in the right image is also detected by Viola and Jones algorithm, and identify its region by a sub-image I_R.

B. Landmark position estimation

When we talk about the landmark position estimation, we are looking for the 3D location of these landmark in the environment and not for the pose (position and orientation) of the object found. To do this task we precalculated the depth using the disparity of one object fixed point.

Getting the landmark identified in both images, we proceed to calculate the points of interest. To do this we use Canny edge detection operator on I_L and a correlation. A number of correlation-based algorithms attempt to find points of interest on which to perform the correlation. In fact, the normalization embodied into the *Normalized Cross Correlation* (NCC) and *Zero Mean Normalized Cross Correlation* (ZNCC) allows for tolerating linear brightness variations. Further more, thanks to the subtraction of the local mean, the ZNCC provides better robustness than the NCC [8] since it tolerates uniform brightness variations as well.

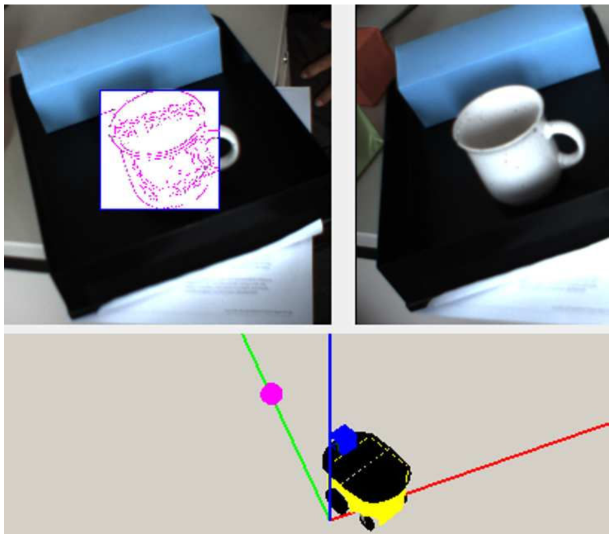

Fig. 3. Identify an object and getting its position in the 3D virtual environment

Correspondences of an image patch are searched for along the epipolar line by calculating the ZNCC only in a given interval $(d_{min}, \ldots, d_{max})$ of so-called disparities [9] [10]. The term *disparity* denotes the Euclidean distance from one point on the epipolar line to a given point in the other camera image [11]. A small disparity represents a large distance to the camera, a large value a small distance (parallax).

When all the points are matched in both images we proceed to calculate its 3D position using the triangulation.

Then we integrate this set of points to get its center of gravity and place the center of a sphere on it. The radius of the sphere is calculated taking the highest number of points of the landmark. The sphere is stored in the 3D virtual map using CGA and it is labeled as a landmark.

VI. Experiments

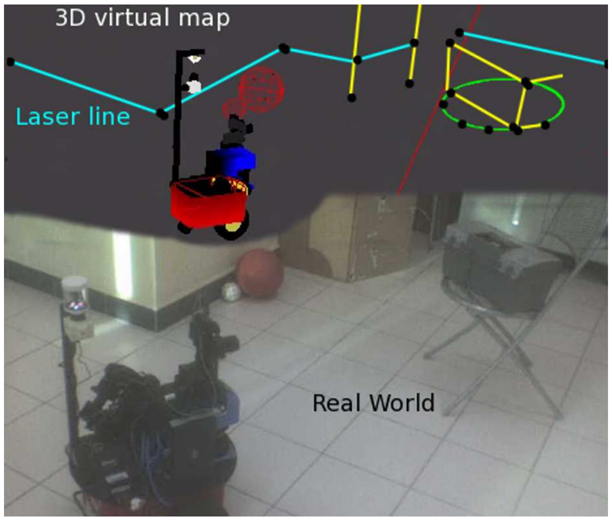

Fig. 4. Registering the environment, merging laser and stereo data

The fig. 1 shows a virtual scenario using a laser rangefinder simulator and a real 3D objects capturing with the stereo camera. Here we can see the 3D objects (the small colored objects) are in the place where the robot took the samples from them (its 3D position), also we can complete the environment representation using the virtual walls.

Fig. 4, shows the creation of a real small 3D map merging the data registered by the laser rangefinder and the 3D shapes recorded by the stereo camera system. Here we can see that the laser line (line segments) are together with the rest of the 3D objects in the right pose.

In the Fig. 5, it is shown the creation of a real 3D map and the found landmarks.

All the maps are stored using only CGA entities, this help to reduce storage space and it has all the information about the objects inside the environment, and it has the property to be updated. Also we can apply any transformation as *rotation* or *translation* to move on the map.

VII. Conclusions

In this paper the authors have shown the use of geometric entities in Conformal Geometric Algebra (CGA) for modeling input data for a 3D virtual environment, in this way merging in a global coordinate system, the laser rangefinder and stereo camera system (mounted over a pan-tilt unit), Furthermore we present a new method for relocalization using lines and the Hough transform. The machine learning technique is used for the object's recognition. The detected objects are used as a landmarks witch greatly help in the interaction with the environment. The experiments with a real

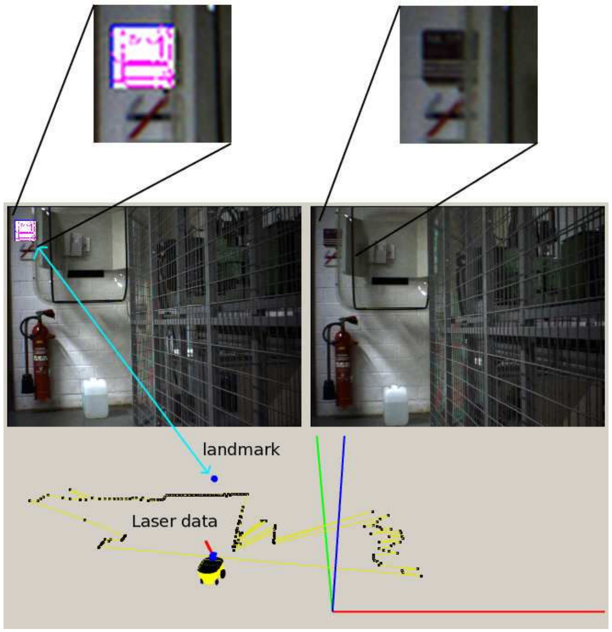

Fig. 5. Mobile robot founding landmarks while it is navigating its environment. On the top see the zoom of the landmark, the stereo view and the 3D map.

robot validate our method. We believe that our approach can be of great use for mobile robots or upper body humanoids installed on mobile platforms.

References

[1] L. Zhang and B. K. Ghosh, "Line segment based map building and localization using 2d laser rangefinder," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, 2000, pp. 2538–2543.

[2] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson, "An optimized segmentation method for a 2d laser-scanner applied to mobile robot navigation," in *Proceedings of the 3rd IFAC Symposium on Intelligent Components and Instruments for Control Applications*, pp. 153-158., 1997.

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

[4] E. Bayro-Corrochano, "Robot perception and action using conformal geometry," in *the Handbook of Geometric Computing. Applications in Pattern Recognition, Computer Vision, Neurocomputing and Robotics*, E. Bayro-Corrochano, Ed. Springer Verlag, Heidelberg, 2005, ch. 13, pp. 405–458.

[5] 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 archive*, vol. 13, pp. 79–100, October 2000.

[6] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d range data for indoor mobile robotics," *Auton. Robots*, vol. 23, no. 2, pp. 97–111, 2007.

[7] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features.," In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 511-518, December 2001.

[8] L. Di Stefano, S. Mattoccia, and F. Tombari, "ZNCC-based template matching using bounded partial correlation," *Pattern Recogn. Lett.*, vol. 26, no. 14, 2005

[9] O. Faugeras et al., "Real-time correlation-based stereo: algorithm, implementation and applications," *INRIA Technical Report no. 2013*, 1993.

[10] P. Azad, T. Gockel, R. Dillmann,"Computer Vision: Principles and Practice," Ed. Elektor Electronics, 2008

[11] R. Hartley, A. Zisserman, "Multiple View Geometry in Computer Vision," Ed. Cambridge University Press, 2004