Construction of Roadmaps Maps for Mobile Robots' Navigation Using RGB-D Cameras

Jesus Savage, Luis Contreras, Israel Figueroa, Abel Pacheco, Marco Negrete, Mauricio Matamoros, Carlos Rivera

Bio-Robotics Laboratory, Universidad Nacional Autónoma de México, UNAM

Abstract.
This paper describes the construction of roadmaps for mobile robots using vector quantization clustering of the free space. A RGB-D camera is used to find planes that represent objects in the environment as well as walls and floors. The free space is found by separating the objects' and walls' planes from the floor's plane, that represents the space where the mobile robot navigates. Vector quantization technique is used to partition the free space into regions, and the centroids of the regions become the nodes of a roadmap, that is used by a mobile robot to navigate.

KEY WORDS

Keywords: Space representations, Roadmaps Maps for Mobile Robots, Vector Quantization Clustering

1 *Introduction*

Roadmaps maps are useful for robots to navigate in structured environments while performing tasks. There are several techniques to build roadmaps for different kinds of problems where a geometrical representation of the objects in the environment (such as rooms and other obstacles) is available: visibility maps[1], Voronoi diagrams [2] and the popular probabilistic roadmap methods (PRM) [3]. In problems where sensors readings are the only information to build a map of the free space, a roadmap can be built using Voronoi diagram techniques. In this approach, a Voronoi diagram separates the free space into regions whose centers are the nodes of the roadmap.

This paper describes a method to create roadmaps using vector quantization (VQ) with RGB-D sensor raw data, obtained by Microsoft Kinect device, mounted in a mobile robot. The resulting roadmap has Voronoi characteristics similar to the one proposed in [4].

2 *Cartographer*

The ViRbot system [5] controls the operation of a mobile robot using several subsystems. Each subsystem of the ViRbot has a specific function that contribute to

the overall behavior of the robot. One of this subsystem is the cartographer which represents the workspace of the robot through raw, symbolic and roadmaps. The cartographer is responsible for the creation of roadmaps, specifically, it creates them by using sensor readings of the environment, obtained using a RGB-D device, Kinect.

The RGB-D cameras provide information through a cloud of points that represent the spatial position of each pixel of the captured image. In this research, the RGB information provided by the camera is not used, for the robot only collects 3D readings, $R = \{r_1, r_2, ..., r_N\}$, $r_j = (x_{screen_j}, y_{screen_j}, d_j)$, where $x_{screen_j}, y_{screen_j}$ represent the pixel location in the captured image and d_j the distance to the objects in line of sight. Then $s_j = (x_j, y_j, z_j)$, which represents the spatial positions of the objects' points relative to the Kinect's plane, is obtained by the following transformation:

$$s_j = \boldsymbol{M} r_j$$

where $\boldsymbol{M}$ represents the matrix of intrinsic parameters of the Kinect camera, found in the Open NI libraries [7].

Additionally, Microsoft Kinect has inclination sensors that allow reliable inclination measurement to calculate the angle θ of the optic axis Z_o and the vector $\bar{C}$ associated with the force of terrestrial attraction. Using this angle and the height h_k – it is, where the Kinect is located in the robot with respect to the ground, as shown in figure 2 – we can clusterize the cloud points into several planes, as those parallel and perpendicular to the navigation ground.

Figure 1 shows the horizontal and vertical planes found after 3D data collected by the robot.

Fig. 1: 3D scanning of a room, the left side shows a picture of the room to be scanned, the right side shows the horizontal and vertical planes of it, found using the 3D cloud points obtained by a Kinect device.

Fig. 2: Robot Justina with one Kinect device in the top of its mechatronic head.

In this work, with the $S = (s_0, s_1, ..., s_N)$ spatial points, the vertical and horizontal planes of the environment are found by the following procedure: for each pixel r_j a set of four screen points $k_i s$, the so called patch, and their corresponding s_i are found:

$k_1 = (x_{screen_{j-\delta_x}}, y_{screen_{j+\delta_y}})$
$k_2 = (x_{screen_{j+\delta_x}}, y_{screen_{j+\delta_y}})$
$k_3 = (x_{screen_{j+\delta_x}}, y_{screen_{j-\delta_y}})$
$k_4 = (x_{screen_{j-\delta_x}}, y_{screen_{j-\delta_y}})$

Where δ_x and δ_y are the displacements in the x and y axis respectively.

Then, using the $s_i, i = 1, 2, 3, 4$, corresponding to the $k_i s$, a normal vector is calculated by the cross product of vectors $\bar{A}$ and $\bar{B}$, see figure 3. These vectors are:

$$\bar{A} = [(x_3 - x_1), (y_3 - y_1), (z_3 - z_1)] = [a_x, a_y, a_z]$$

and

$$\bar{B} = [(x_2 - x_4), (y_2 - y_4), (z_2 - z_4)] = [b_x, b_y, b_z],$$

Thus the normal $\bar{N}_j$ in S_j is

$\bar{N}_j = \bar{A}\boldsymbol{X}\bar{B}$
$= [(a_y * b_z - b_y * a_z), (a_z * b_x - b_z * a_x), (a_x * b_y - b_x * a_y)]$

The cloud points can be label according to which plane they belong [6], in our case we were interested only in the vertical and horizontal ones. Thus, using the normal $\bar{N}_j$ of point S_j and the matching vector $\bar{V}_h = [010]$, relative to the horizon, to detect if belongs to an horizontal or vertical plane by:

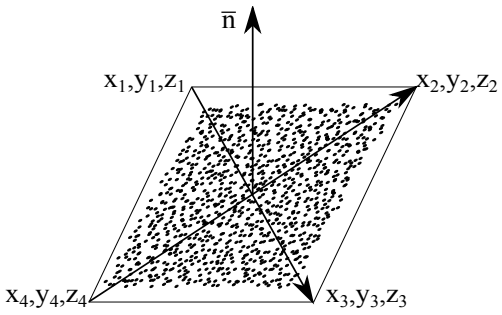

Fig. 3: Normal vectors are found using 4 points in the neighborhood of $s_j = (x_j, y_j, z_j)$.

$cos(\alpha) = \frac{\bar{N}_j \cdot \bar{V}_h}{|\bar{N}_j||\bar{V}_h|}$

where α is the angle between vectors $\bar{N}_j$ and $\bar{V}_h$. if $\alpha \approx 0$ then $\bar{N}_j$ belongs to an horizontal plane, if $\alpha \approx 1$ then $\bar{N}_j$ belongs to a vertical plane.

Then given the $S = (s_0, s_1, ..., s_N)$, where $s_j = (x_j, y_j, z_j)$, they are mapped to the ground map, as shown in figure 4, by the following transformation:

$t_j = \boldsymbol{M}_T s_j = [xg_j, yg_j, zg_j]$

where $\boldsymbol{M}_T$ is the transformation matrix:

$$\boldsymbol{M}_T = \begin{vmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sen(\theta) \\ 0 & sen(\theta) & cos(\theta) \end{vmatrix}.$$

Then, $T = (t_0, t_1, ..., t_N)$ represents all the points grounded in the navigation plane, and the free space Q_T is the set of points in T that belong to horizontal plane where the robot navigates. The cloud points of the horizontal planes, higher than the ground plane, and the vertical planes that represent the objects and walls are projected to the planed where the robot navigates, and they are consider as the non navigable space Z_T. Thus, Q_T the free space and Z_T the occupied one, are obtained from T.

Finally, we apply a perspective transformation from plane XY to plane XZ. Figure 5 shows these planes separation, the blue color represents the navigable space, the pink represents the horizontal occupied one, while the brown represents the vertical planes, the yellow represents the unknown space, that is, data that is neither in a vertical nor in an horizontal plane and finally the gray color represents the space where no data were collected by the 3D sensing device. In this figure it is shown an empty space in the middle of it, this represents a

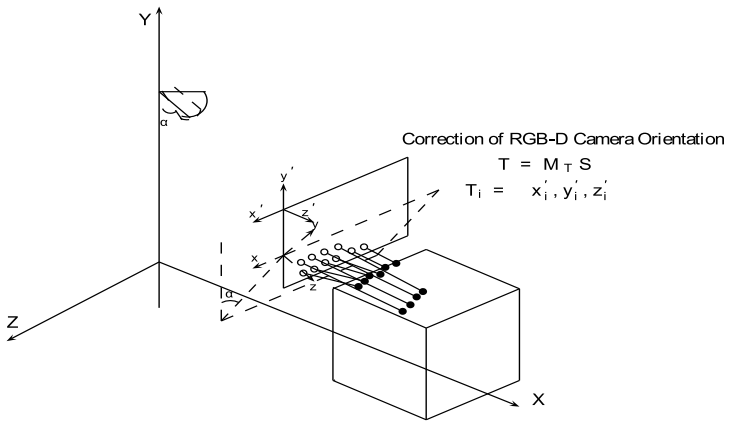

Fig. 4: The horizontal and vertical planes are projected to an horizontal plane where the robot navigates.

shadow of the box, shown in the left side of figure 1, that was generated by the laser beam of the Kinect device.

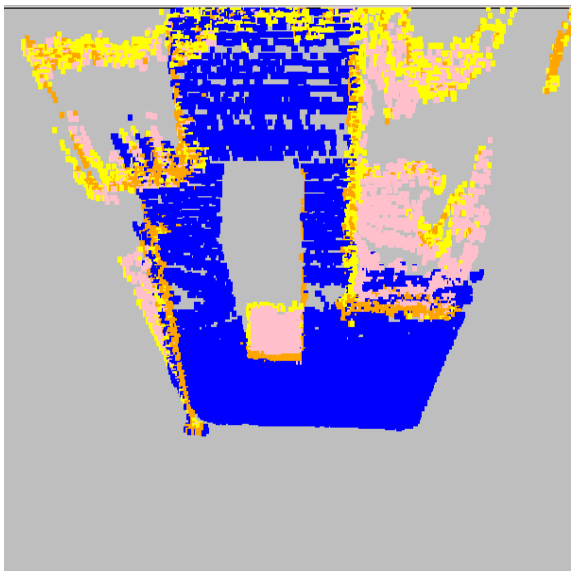

Fig. 5: Projections of the horizontal and vertical planes into the navigation plane.

We summarize this previous procedure as follows:

1. Capture the RGB-D information and get the spatial position per pixel $S = (s_0, s_1, ..., s_N)$.

2. Define a patch around each pixel and calculate the normal $\bar{N}_j$ in S_j.

3. Rotate the normal $\bar{N}_j$ according to the correction of the camera orientation for referencing it to the canonical horizon.

4. Calculate the projection of $\bar{N}_j$ in $\bar{V}_h$, to determinate if S_j belong to an horizontal or vertical plane and associate them to $T = (t_0, t_1, ..., t_N)$.

5. Apply a perspective transformation from plane XY to plane XZ to have an aerial point of view.

Using the points in T, that belong to Q_T and Z_T, and vector quantization techniques a set of centroids corresponding to the the free space and the occupied one are found. The centroids of Q_T are in turn nodes of the roadmap of the free space. The following section explains how to use vector quantization techniques to achieve this purpose.

2.1 Vector Quantization

Vector Quantization techniques [8] have been extensively used for data compression in digital signal processing and telecommunications. For mobile robot applications, VQ is also used for data compression and to find regions in the environment.

Given a set of N_v vectors, $q_j = \{x_j, y_j, z_j\}; j = 1, ..., N_v$ that represent the position of points in the free space, a set of centroids which represents these vectors is found.

A collection of centroids is called a codebook. The codebook is designed from a long training sequence that is representative of all vectors q_j to be encoded by the system. The codebook is created with the Linde-Buzo-Gray (LBG) algorithm [8], which is based on the generalized Lloyd Algorithm [9].

The LBG algorithm is as follows:

1. Find an initial codebook D_1, with one centroid C_1, by averaging all the vectors q_j, with $L = 1$ and $m = 1$.

2. Modify each of the centroids $C_i; i = 1, ..., L_m$ in D_m by adding them a vector $\pm\psi$ of small magnitude to generate 2 new centroids from each of them, generating a new codebook D_{m+1}, and $L_{m+1} = 2 * L_m$, $m = m + 1$.

3. Given a codebook $D_m = C_i; i = 1, ..., L_m$ assign each vector q_j into the clusters R_k whose centroid C_k is closer to q_j according to some similitude measurement $d_j = d(q_j, C_k)$. The measurement used is the Euclidean distance between two points.

4. Recompute the centroids C_k for each of the clusters, R_k, by averaging all the vectors q_j that belong to R_k.

5. If the difference between the average distance $\bar{d}_t = \frac{1}{N_v}\Sigma d(q_j, C_k)$, in iteration t, between vectors q_j and their corresponding centroids C_k, and the previous average distance $\bar{d}_{t-1}$, $|\bar{d}_t - \bar{d}_{t-1}| \geq \epsilon$, go to 3.

6. If $L_m <$ codebook size go to 2.

Where codebook size is the number of regions of the environment. This number is limited by the time of computation for real time operation, figure 6 shows in the left side the vertical and horizontal planes found in an office, in the middle it is shown these planes mapped to the ground and the right side shows the VQ regions of the free space, in color, and occupied space, black color.

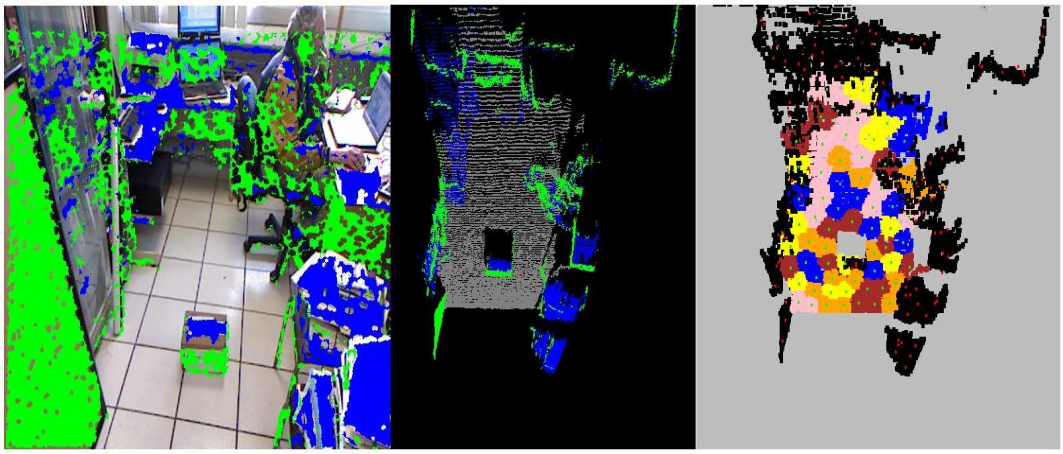

Fig. 6: In the right side it is shown the partitions of the free space Q_T, in an office environment, the green points represent their centroids, while the black regions represent the occupied space Z_T, and the red dots their corresponding centroids.

2.2 Roadmap Construction Algorithm

Finally the roadmap is build following the next algorithm:

Input:

N: number of nodes in the roadmap, it should be a power of 2.
P: centroids of the vector quantizer of the occupied space Z_T.
C: centroids of the vector quantizer of the free space Q_T.

Output:

A roadmap $G = (V, E)$
V: Nodes of the roadmap
E: Edges of the roadmap

1: $E \leftarrow 0$
2: $V \leftarrow C$
3: for all $v \in V$ do
4: for all $v' \in V$ do
5: if the edge of $(q, q') \notin$ E and Vis$(q, q', p) \neq NIL, \forall \ p \in P$
6: $E \leftarrow E \ \cup$ edge of (q, q')
7: end if
8: end for
9: end for

$Vis(q, q', p)$ is a function that detects if the edge that connects q and q' collides or its to close with a centroid p in Z_T. Figure 7 shows two centroids of the free space $q_1 = (x_1, y_2)$, $q_2 = (x_2, y_2)$ and a centroid $p = (x_p, y_p)$ in the occupied one. Figure 7 show this configuration where a is the magnitude of the line that joins free space centroids q_1, q_2, $a = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, b is the magnitude of the line that joins the free space centroid q_2 and the occupied one q_p and c is the magnitude of the line that joins the free space centroid q_1 and q_p. Then the distance l from the line that joins q_1, q_2 and p is:

$$l = c \ \ sen(\theta)$$

$$\theta = ang \ \ cos(\frac{a^2 + c^2 - b^2}{2 \ \ a \ \ c})$$

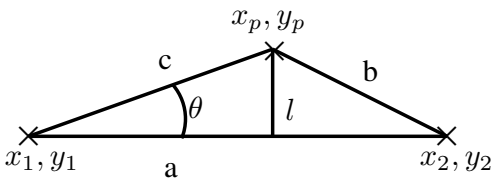

Fig. 7: Function $Vis(q, q', p)$ detects a collision with a centroid in the occupied space if $l < \epsilon$

Thus, if $l < \epsilon$ then the function $Vis(q, q', p)$ detects a collision with a centroid of the occupied space and the centroids in the free space q, q' thus and edge between $Vis(q, q', p)$ is not created. ϵ is proportional to the robot's radio.

3 *Experiments and Results*

The system was tested in different environments and conditions. Firstly, we were interested on the number of regions for both the free and occupied space necessary to find a suitable roadmap for navigation. The parameters of the experiments are:

1. Number of centroids L_Q in the free space.
2. Number of centroids L_Z in the occupied space.
3. Largest distance δ_c of the edges between nodes of the roadmap.

Figure 8 shows the partition of the free space with 16 centroids and the distance between edges $\delta_c = 0.3m$, as it is shown in the left side of it, there is not a suitable roadmap for a robot mobile to navigate. In the right side it shown the same partition of the free space but now with a distance $\delta_c = 0.4m$

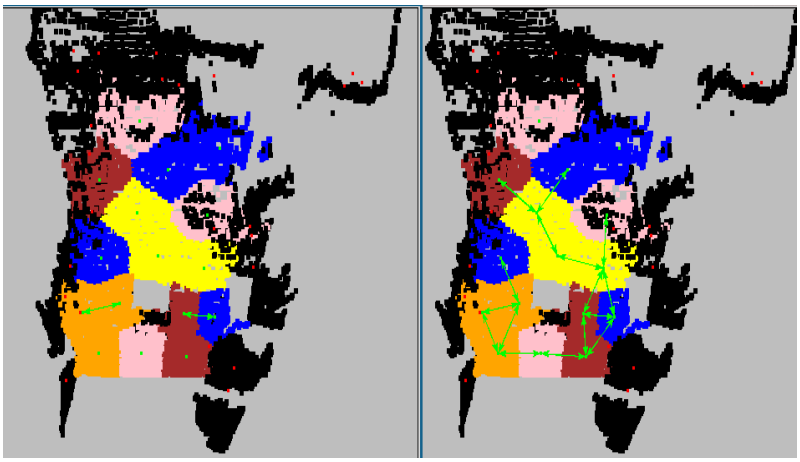

Fig. 8: The left side shows a VQ with 16 centroids in the free space and a maximum distance between the nodes of 0.3 meters, while the right side shows the same configuration with a distance of 0.4 meters.

Figure 9 shows the best path from an origin to a destination using the roadmap shown in the right side of figure 8 and the Dijkstra algorithm, the red circles represent the robot's positions.

Figure 10 shows in the left side the partition of the free space with 32 centroids and the right side with 64 centroids with a maximum distance between the nodes of 0.3 meters.

As we can see in figure 11, the path with the same origin and destination as in 9 is more efficient because it occupies less steps to reach the destination.

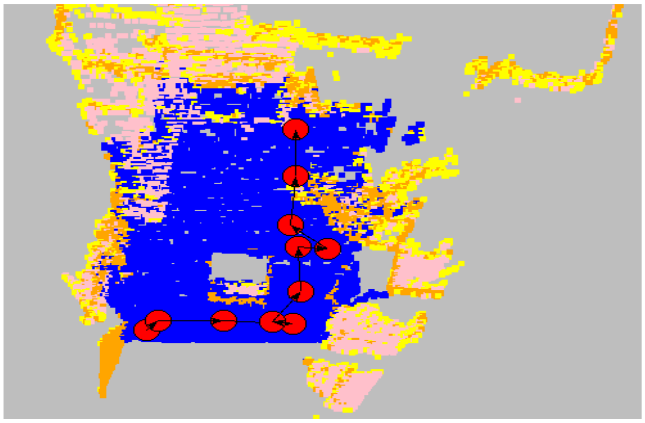

Fig. 9: Path with a roadmap created with a 16 centroids VQ and a max 0.4 m distance between them.

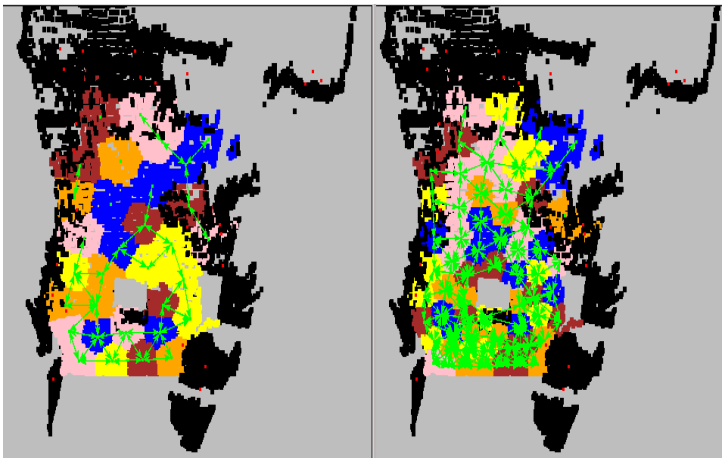

Fig. 10: The left side show VQ of the free space with 32 centroids and a maximum distance between centroids of 0.3 meters, the right side show a VQ of 64 centroids with the same distance.

Table 1 shows the results of the experiments, randomly for each configuration were selected 10 origins, x_o, y_o, and 10 destinations x_d, y_d in the free space. One of the columns of this table shows the number of times a path was found from the origin and the destination using the roadmap created and the Dijkstra algorithm. It was measured also the efficiency of the system according to the following criteria:

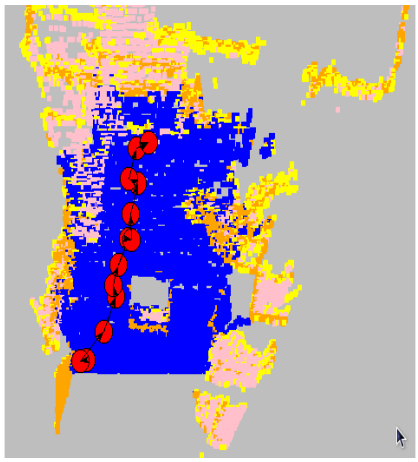

Fig. 11: Path with a roadmap created with a 64 centroids VQ and a maximum 0.3 m distance between them.

$$Ef = DistDest + nSteps + CmpTime + Path$$

Where $DistDest$ is the distance between the last position of the robot and the destination; $nSteps$ is the number of steps used to reach the goal; $CmpTime$ computational time; $Path$ evaluates the smoothness of the paths, that is related to the angles'size of the rotations between segments of the robot.

Table 1 shows the relative efficiency Ef_r of each configuration to the highest one of the overall experiment.

L_q	L_z	$delta_c$	# Times Reached Destination	Relative Efficiency Ef_r
16	16	.3 m	2	0.12
16	16	.4 m	4	0.44
32	16	.3 m	8	0.72
32	16	.4 m	8	0.77
32	32	.4 m	8	0.86
64	32	.3 m	10	1.00
64	32	.4 m	10	0.94
64	64	.3 m	10	0.83
64	64	.4 m	10	0.82

Table 1: Results of the creation of roadmaps using VQ

From table 1, it can be inferred that the best configuration is with 64 centroids in the free space, 32 in the occupied space and a distance between nodes $\delta_c = 0.3$.

4 *Conclusions*

In this paper was described the construction of roadmaps for mobile robots by making clusters of the free space given a cloud of points obtained by a RGB-D device, the Kinect. Using this technique a roadmap, similar to the one developed using Voronoi diagrams, was created, but with more flexibility for choosing the size and the number of regions in the environment. The resulting roadmaps are used by our mobile robot, Justina, to navigate in structured environments, as the one presented in the category @Home at the RoboCup [10].

ACKNOWLEDGMENT

This work was supported by PAPIIT-DGAPA UNAM under Grant IN-107609

References

1. Lozano-Perez, T. and Wesley, M.A. *An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles.* Communications of the ACM, 22(10),560-570, 1979.
2. Jean-Claude Latombe. *Robot Motion Planning.* Kluber Academic Publishers Group, 1991.
3. L.E. Kavraki, P. Svetska, J.C. Latombe, and M.H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces.* IEEE Transactions on Robotics and Automation, 12(4):566-580,June 1996.
4. Thrun, Sebastian, and Bucken, Wolfram Burgard, Dieter Fox. *Map Learning and High-Speed Navigation in RHINO.* Artificial Intelligence and Mobile Robots. MIT Press, 1998.
5. J. Savage, Adalberto LLarena, Gerardo Carrera, Sergio Cuellar, David Esparza, Yukihiro Minami, Ulises Penuelas; *ViRbot: A System for the Operation of Mobile Robots.* RoboCup 2007 Symposium, Atlanta, EU, 2007.
6. D. Holz, S. Holzer, R. B. Rusu, and S. Behnke (2011, July). Real-Time Plane Segmentation using RGB-D Cameras. In Proceedings of the 15th RoboCup International Symposium, Istanbul, Turkey.
7. OpenNI. Openni. http://www.openni.org, 2009.
8. Y. Linde, A. Buzo, and R. M. Gray. *An Algorithm for Vector Quantizer Design.* IEEE Transactions on Communications, 84-95, 1980
9. S.P. Lloyd.*Least Squares Quantization in PCM.* IEEE Transactions on Information Theory, IT-28: 127-135, 1982.
10. http://www.robocupathome.org/