CHAPTER X

A COMPLETE 3D PERCEPTION AND PATH PLANNING ARCHITECTURE FOR A HUMANOID

M. GONZÁLEZ-FIERRO, J.G BUENO, C. BALAGUER y L. MORENO

Robotics Lab, Universidad Carlos III de Madrid

In this paper, a perception and path planning architecture is presented for HOAP humanoid. The main goal of this study is to integrate several researches in navigation and perception and create a full statement problem so the robot is able to interpret the surroundings with a RGB-D sensor and create a valid path planning to achieve the desired goal.

1 Introduction

If robots are going to share our living space, they should be able to move in clustered environments with reliability, avoiding fixed and moving obstacles. Smart environments are an easy solution to this matter (Pierro, 2012; Víctores, 2010; Coradeschi, 2006). The main objective of a smart environment is to reduce the complexity of a determined task and to help the robot to perform this task. First, by installing external cameras or sensors, a global and more accurate view of the environment can be perceived. It can help the robot to free computational resources by eliminating the need of doing SLAM. Also, a smarter high-level action planning can be performed by anticipating the location of fixed or moving obstacles. Furthermore, it reduces the dependence of the inner robot sensors.

In this paper we present a framework to generate a safe path planning pattern in a humanoid robot, by perceiving a complete 3D map of the environment using an external RGB-D sensor, in the context of smart environments. First, a 3D model of the environment is extracted by identifying the free space and non-dynamic obstacles. Using this model, a Bi-RRT path planning algorithm is computed to obtain a first safe path of the Center of Gravity (COG) of the robot. Using this preliminary solution and taking into account the dimensions of the robot, a Zero Moment Point (ZMP) patter is generated. Next, using the cart-table algorithm, we can compute the real COG trajectory which is used to generate the locomotion.

The paper is ordered as follows. In section 2 the 3D model of the environment is obtained and the perception procedure is explained. In section 3 we address the path planning algorithm. In section 4 we discuss the proposed architecture. Finally, in section 5 we show the experimental results with a simulated and real robot and in section 6 we present the conclusions.

2 Perception of the environment

In this section will be explained all the theoretical principles used to integrate the perception feature into the humanoid architecture.

2.1 Environment perceptions

The very first goal of the robot is to understand the surroundings in order to construct a valid path and therefore achieve successfully the desired goal. For this study, a RGB-D Asus Xtion Pro Live camera has been installed in the upper part of a room with a certain angle. The role of the camera is to acquire a valid model of the supporting plane (floor) and subtract all the possible obstacles that could interfere during the path planning.

The sensor provided by the RGB-D manufacturer is ready to detect indoor 3D points where errors increase quadratic from a few millimeters at 0.5 m distance to about 4 cm at the maximum sensor range (Kourosh, 2012). The camera resolution is VGA (640 × 480 pixels) and output video frame rate may vary between 25 and 30 Hz. Camera provides for each frame a depth map, color map and infrared map. Infrared information has been used on previous researches to fix lens distortions and other optical aberrations by means of chessboard borders detection (Bouguet, 2008) but for this project the standard distortion matrix will be used to simplify the problem and therefore only color and depth sources are used to determine the architecture of the surroundings.

The complete camera flow acquisition is programmed using OpenNI library (OpenNI, 2012) while cloud operations are done using Point Cloud

Library (Rusu, 2009) with *adhoc* interface for the visualization. All the architecture follows a modular structure to facilitate the addition of new filters or improvements.

2.1.1 Down-sampling the data

Camera provides approximately 8M points per second that have to be managed somehow. Due to the significant amount of data that has to be interpreted, it is necessary to reduce the order of magnitude rejecting points in order to let the robot react in real time. In this case, a VoxelGrid filter approach is used. A VoxelGrid represents a small 3D box in space. All the points inside a VoxelGrid are approximated to the centroid reducing the amount of data. The bigger the VoxelGrid, the smaller the data obtained and so the faster the operations (see Fig.1).

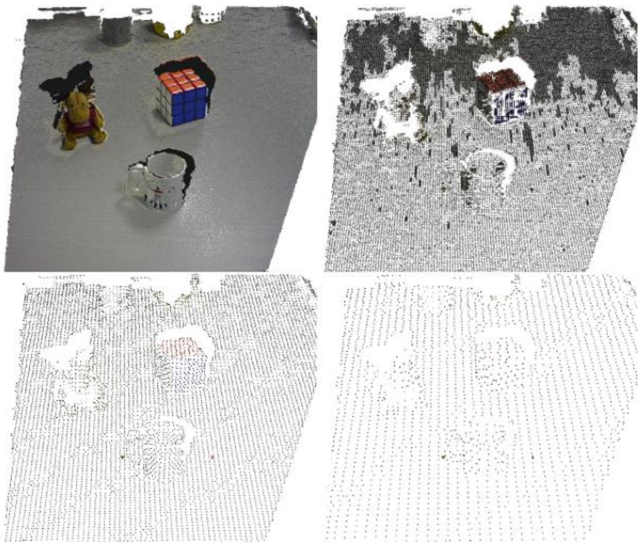

Fig.1. Down-sampling the point cloud using voxelgrids at different sizes. Leaf sizes are 0.005m, 0.01m, 0.05m and 0.08m respectively

2.2 Supporting Plane Extraction

In order to determine the surrounding obstacles, the robot has to be able to differentiate between floor and non-floor objects. This distinction has to be done using depth map perception and 3D constraints. In this study, normals extraction is first performed and then the supporting plane is acquired.

2.2.1 Normals computation

Surface normals are very useful to understand the geometry of the surface and also to reconstruct and understand the point cloud. Normal extraction means estimating the normal of a plane tangent to the surface. This process can be as simply as computing the cross product between a pair of nearest points for any query point. This technique is not recommended for noisy point clouds because results are inconsistent and extremely variable (Tamal, 2005).

In this paper, normal extraction has been stated as a least-square plane fitting estimation problem. With the analysis of the eigenvectors and eigenvalues of a covariance matrix C created from the nearest neighbors to the query point, it is possible to determine the surface normal

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \overline{p})\cdot(p_i - \overline{p})^T, C\cdot\overrightarrow{v_j} = \lambda_j\cdot\overrightarrow{v_j}, j \in \{0,1,2\}$$

where λ_j represents the *j-th* eigenvalue and $\overrightarrow{v_j}$ is the *j-th* eigenvector of C. Normals orientation is solved applying the viewpoint constraint given by

$$\overrightarrow{n_i}\cdot(v_p - p_i) > 0$$

Taking in mind that the primary surface direction is given by the third eigenvector

$$\overrightarrow{n_i} = \pm\overrightarrow{v_3}$$

2.2.2 Plane estimation

Once the normal point cloud is computed for each frame, next step is to determine the plane which fits better in the point cloud. To perform this operation RANSAC (Random Sample Consensus) has been applied with a geometric model of a plane. RANSAC will iterate over the whole point population and estimate the parameters of the most suitable plane. To do this, the method distinguishes between inliers (points that fit the model) and outliers (points outside the model) iteratively. For this study, 70% of points are required to be part of the floor and the rest are supposed to be obstacles.

The output of the plane estimation are the classical four parameters that generates a plane equation

$$Ax + By + Cz + D = 0$$

With this information in mind, it becomes straightforward to split up the original depth map into two maps: the first one containing points inside the supporting plane and the other one gathering all the obstacles around the robot.

2.3 Clustering Segmentation

The last step in the perception architecture consists on converting the outliers of the previous section into bounding boxes. Those groups are then introduced into the path planning algorithm as geometric constraints. In order to track each obstacle, a clustering process is performed.

2.3.1 Euclidean Clustering

This algorithm is based on K-means using a *Kd-tree* to boost the performance of the process. It is similar to flood-fill algorithm in image processing. The followed steps are stated in the following algorithm.

```
EUCLIDEAN CLUSTERING
1. **input** point cloud data set $P$
2. **build** Kd-tree ( $P$ )
3. **empty** list of clusters $C$
4. **empty** queue of points to be checked $Q$
```

```
5.  for each $p_i$ in $P$
      add $p_i$ to $Q$
      for each $p_i$ in $Q$
        $P_i^k$ = search $p_i^k$ of neighbours of $p_i$ with $r < r_{threshold}$
        for each $p_i^k$ in $P_i^k$
          if not $Q$ contains $p_i^k$
            $Q \leftarrow p_i^k$
        $C \leftarrow Q$
6.  check $p_i$ in $C$
```

3 Biped locomotion generation

We used the cart-table model (Kajita, 2003) to generate the gait. This model is based on ZMP a preview control scheme to obtain the COG trajectory from a defined ZMP trajectory. This method generates a dynamically stable gait trajectory using the 3D Linear Inverted Pendulum Model (Kanehiro, 2001) to approximate the dynamics of the humanoid.

The relationship between ZMP trajectory and center of gravity (COG) trajectory is defined using the following equations

$$p_x = x - \frac{\ddot{x}}{g} z_c$$

$$p_y = y - \frac{\ddot{y}}{g} z_c$$

Where p_x is the ZMP reference, x is the COG trajectory, $\ddot{x}$ represents the COG acceleration, z_c the COG height and g the gravity.

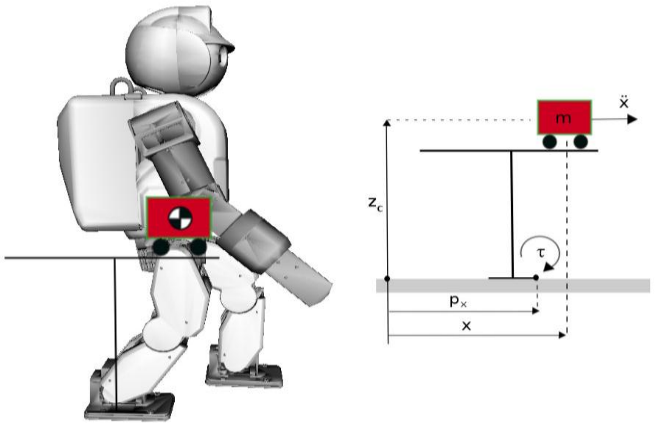

Fig.2. Cart table model in sagital plane

In cart table model (Fig. 2), the cart mass corresponds to the center of mass of the robot. If the cart accelerates at a proper rate, the table can be upright for a while. At this moment, the moment around p_x is equal to zero, so the ZMP exists.

$$\tau_{ZMP} = mg(x - p_x) - m\ddot{x}z_c = 0$$

3.1 Path Planning

Two of the most successful algorithms include Probabilistic Roadmap Method (PRM) (Kavraki, 1996) and Rapidly-exploring Random Tree (RRT) (LaValle, 1998). We used the Rapidly-Exploring Random Trees Algorithm. Bi-directional RRTs (Kuffner, 2000).

The Rapidly-exploring Random Tree (RRT) was introduced in (LaValle, 1998) as an efficient data structure and sampling scheme to quickly search high-dimensional spaces that have both algebraic constraints (arising from obstacles) and differential constraints (arising from non-holonomy and dynamics).

The key idea is to bias the exploration toward unexplored portions of the space. In the current paper, we present an approach that is tailored to problems in which there are no differential constraints, and the problem can be expressed in *C-Space*. The basic RRT construction algorithm is explained in Fig. 3.

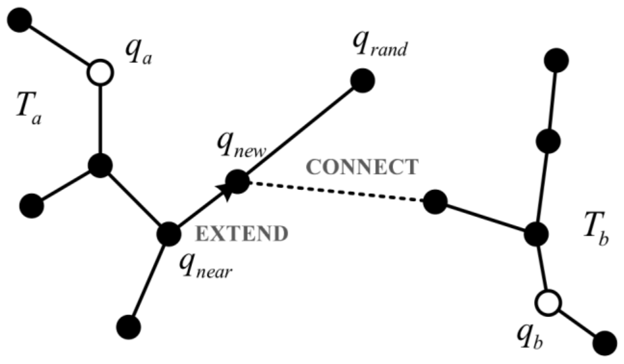

Fig.3. Process of RRT-Connect merging two RRT trees

As illustrated in Fig.3, the RRT-Connect works by extending and connecting two trees toward each other. Two trees (T_a and T_b), rooted at two different milestones (q_a and q_b), either be local tree or global tree, are maintained at all times until they are connected to each other and merged into one single RRT.

At every time step, a random configuration q_{rand} is sampled inside the free space C_{free}. The **EXTEND** function determines the nearest configuration of q_{rand} in the current tree T_a, denoted as q_{near}. After that, T_a extends in the direction of q_{near} for one step, generating a new configuration q_{new}, using a fixed incremental distance.

From this point three situations can occur:

1. *Reached*: $q_{new} = q_{rand}$ and the configuration is directly added to the tree because it already contain a vertex within a fixed distance of q_{rand}
2. *Advanced:* $q_{new} \neq q_{rand}$ and it is added to the tree
3. *Trapped*: new configuration is rejected because it does not belong to the free space C_{free}.

At the same time, a tree T_b starts to grow from the goal configuration q_b. The other tree T_b uses another procedure called **CONNECT** to extend toward q_{new} as much as possible. The **CONNECT** procedure is a greedy function that can be considered as an extension of the **EXTEND** function. Instead of attempting to grow towards the sample q_{random}, it iterates the **EXTEND** function towards q_{new}, until a configuration is reached or there is an obstacle.

If the T_b can successfully reach q_{new}, the two trees are connected and merged into one single RRT. It is all explained in the following algorithms.

EXTEND (T, q)

```
1. $q_{near} \leftarrow Nearest.Neighbour(q,T)$
2. if new_config ($q, q_{near}, q_{new}$) then
     | add_vertex ($q_{new}$)
     | add_edge ($q_{near}, q_{new}$)
     if $q_{new} = q$ then
       return reached
     else
       return advanced
3. return trapped
```

BUILD RRT

```
1. init $T(q_{init})$
2. for $k = 1$ to $K$ do
     | $q_{rand} \leftarrow Rnd.Cfg$
     | extend($T, q_{rand}$)
3. return $T$
```

CONNECT (T, q)

```
1. do
     $S \leftarrow$ extend($T, q$)
2. until not ($S = advanced$)
3. return $S$
```

RRT CONNECT PLANNER (q_{init}, q_{goal})

```
1. init $T_a(q_{init}), T_b(q_{init}, q_{goal})$
2. for $k = 1$ to $K$ do
     | $q_{rand} \leftarrow Rnd.Cfg$
     | If not extend($T_a, q_{rand}$) == $trapped$ then
         If connect($T_b, q_{new}$) == $reached$ then
           Return path($T_a, T_b$)
       swap($T_a, T_b$)
3. return failure
```

In every iteration, one tree is extended to the new configuration q_{new} and the other attempts to connect its nearest branch to the other tree reaching q_{new}. Then, the roles are reversed by swapping the trees. This causes both trees to explore the free space while attempting to connect each other

4 Proposed architecture

In Fig. 4 a representation of the complete system is presented. The location of the RGB-D sensor is selected to cover the working from the top-view upper part of the room.

The first step is to extract the supporting plane equation from the Point Cloud. Afterwards, the objects are identified using Euclidean Clustering Algorithm, and the equation of the bounding box containing each object is obtained. The plane coefficients and the bounding box of every object and its position are written to a *xml* file, which defines the 3D model of the environment.

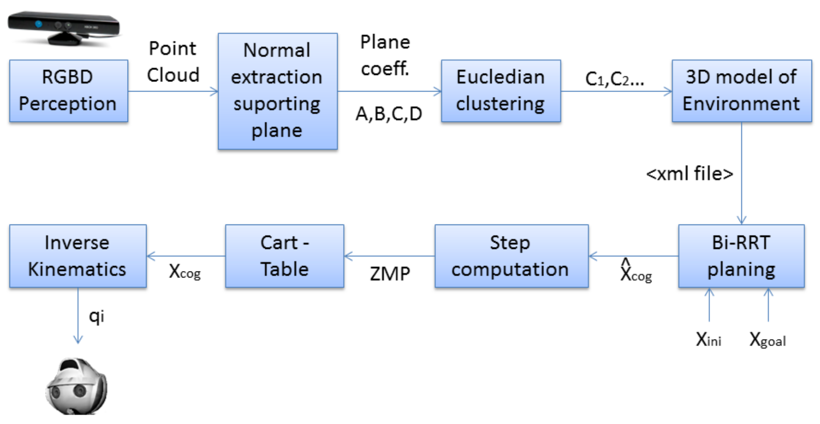

Fig.4. Schema of the proposed architecture. The output of each step is represented within the arrows.

We select the initial and final position of the walking trajectory, which in addition to the 3D model of the environment, is the input to the Bi-RRT algorithm. The result of this path planning is a preliminary cartesian COG trajectory $\hat{X}_{COG}$. The output of the Bi-RRT algorithm can be seen in the following Fig. 5. The initial point is located near the bottom of the scenario and the final point is near the front door. To configure the free space, the Bi-RRT algorithm not only takes into account the data of the xml file, but also the dimensions of the robot plus a security quantity.

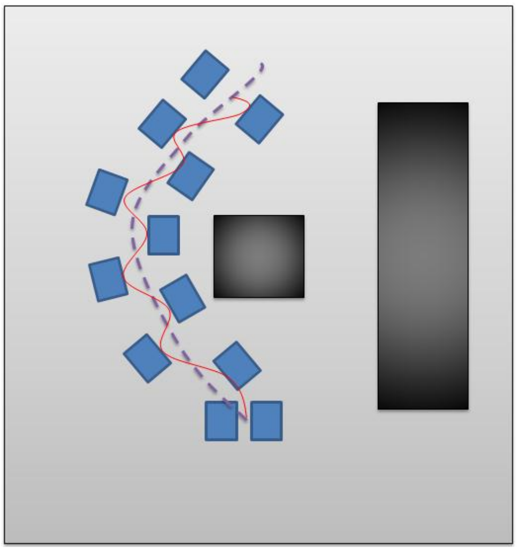

Fig.5. Top-view schema of the complete scenario with obstacles in dark gray, global path planning represented by the dotted line, steps plotted as blue boxes and finally COG path shown with a red thin line.

Finally, from the cartesian COG trajectory X_{COG}, the articular trajectory of every humanoid joint q can be computed, using a kinematic inversion.

5 Experimental results

In this section the experimental results are presented and discussed. In Fig.6 a simulation of the environment is shown using OpenRAVE (Diankov, 2010). Objects identified by the Asus camera are represented as light grey boxes. The first one is ahead the robot, and the second one is to its right. The floor is represented in dark grey. The rest of the environment have been manually introduced and is irrelevant to the experiment. The yellow dotted line represents the field of view of the RGB-D camera installed in the upper part of the scenario.

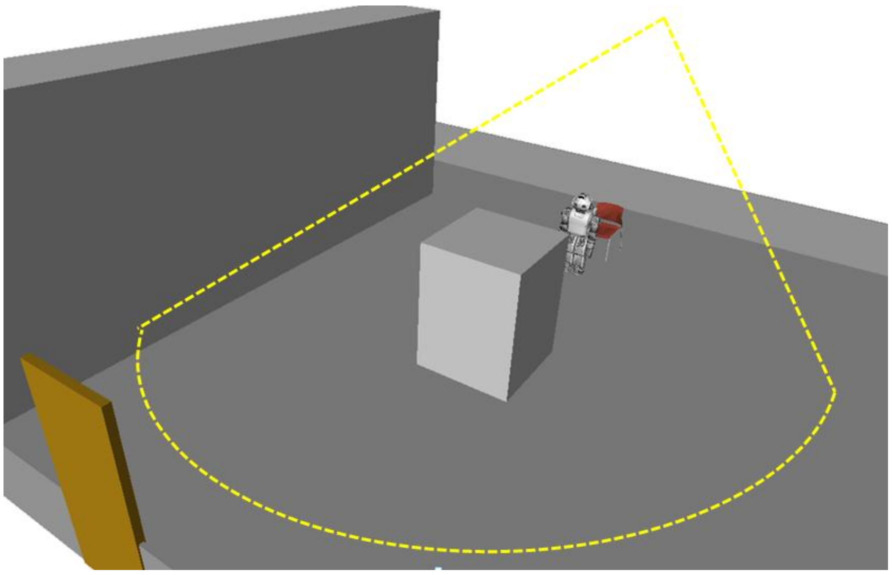

Fig.6. Simulation of the environment with bounding boxes representing obstacles detected by the RGB-D sensor. Yellow dotted line represents the field of view of the RGB-D sensor.

The result of the path planning algorithm integrating the data obtained with the perception sensor is successfully represented in Fig.7 where the HOAP's head represents the COG of the complete robot and the plotted trajectory the final COG path estimation.

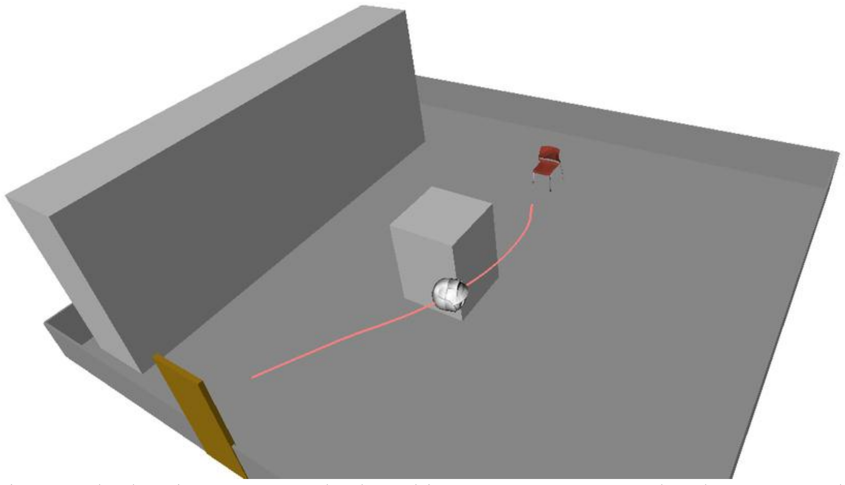

Fig.7. Path planning COG result plotted in OpenRave. HOAP head represents the COG of the complete robot.

In Fig. 8 a group of snapshots of the real humanoid performing the navigation is shown. The robot performs a safe locomotion trajectory avoiding the obstacles in the environment.

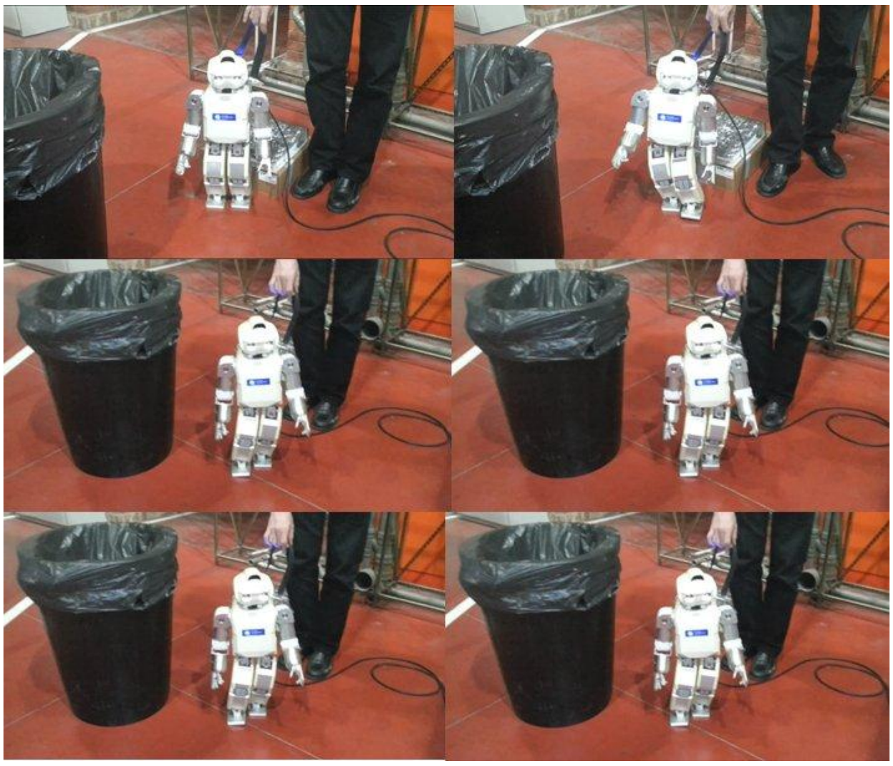
Fig.8. Several frames representing the real performance of the robot path planning in the laboratory. The black bucket is one of the obstacles avoided.

6 Conclusions

In this paper we present a humanoid robot performing a safe path planning in a cluttered environment. Instead of using the cameras of the robot to create a model of the environment, we used an external fixed camera. The RGB-D sensor is located in a position where a large area of the environment can be analyzed. We obtained a 3D model of the floor and the obstacles to define the free space using a perception technique that includes a supporting plane extraction and object clustering. Once the 3D model of the environment has been computed, a safe path using Bi-RRTs has been planned.

This trajectory, which has into account the shape of the humanoid, is used to calculate the ZMP trajectory of the robot. Using the cart-table algorithm, we obtain the humanoid COG trajectory, which allows the computation of the locomotion pattern. The result is a safe locomotion trajectory that allows the robot to avoid obstacles.

6.1 Future directions

The next step towards a more autonomous and robust path planning can be a combination of external sensors with the cameras integrated in the robot. Using this combination, a more accurate model of the environment can be extracted and can even avoid moving obstacles. One of the problems of using the cameras located in the eyes of the robot is the swing movement that appears when the robot is moving. The cameras that suffer from a *cuasi*-horizontal movement produce a blurry image. This image has to be treated before it can be used. A solution to this can be a camera stabilizer algorithm like (Mateo, 2010).

References

Bouguet, J. Y. (2008). *Camera Calibration Toolbox for Matlab*. Available from `http://www.vision.caltech.edu/bouguetj/calib/doc/index.html`

Coradeschi, S.; Saffiotti, A. (2006) *Symbiotic Robotic Systems: Humans, Robots, and Smart Environments*, Intelligent Systems, IEEE , vol.21, no.3, pp.82-84. doi: 10.1109/MIS.2006.59

R. Diankov, *Automated Construction of Robotic Manipulation Programs* (2010) PhD. Thesis Carnegie Mellon University, Robotics Institute.

S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. *Biped walking pattern generation by using preview control of zero-moment point* (2003). In Robotics and Automation,. Proceedings. ICRA'03. IEEE International Conference, vol. 2. IEEE, pp. 1620–1626.

L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, (1996) *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, IEEE Trans. Robot. & Automat., pp. 12(4):566–580.

F. Kanehiro, K. Kaneko, K. Yokoi, H. Hirukawa, and S. Kajita, (2001) , *The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation*, in Proceedings of the IEEE Conference on Intelligent Robots and Systems, pp.239–246.

J.J. Kuffner and S.M. LaValle (2000), *RRT-Connect: An efficient approach to single query path planning*. In Proc. IEEE Int Conf. on Robotics and Automation (ICRA'2000), pp. 995–1001, San Francisco, CA, April 2000.

Kourosh Khoshelham, Sander Oude Elberink. (2012), *Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications*. Sensor, 12, pp.1437–1454.

S. M. LaValle. (1998) *Rapidly-exploring random trees: A new tool for path planning*. Pp. 98–11, Computer Science Dept., Iowa State Univ.

A.P.Mateo, M.G.Fierro, D.Hernandez, P.Pierro, C.Balaguer, (2010) , *Robust Real Time Stabilization: Estabilización de la imagen con aplicación en el robot humanoide HOAP-3.*7º Wokshop Robocity2030. Visión en robótica. Madrid.

OpenNI (2012), *The largest 3D sensing development framework and community*. http://www.openni.org

P.Pierro, D.Hernandez, D.Herrero, M.G.Fierro, C.Balaguer. (2012) *Perception System for Working with Humanoid Robots in Unstructured Collaborative Scenarios.* Proceedings of the 2012 International IEEE Intelligent Vehicles Symposium. Workshops V Perception in Robotics. Alcalá de Henar. Spain.

Radu, B. R. (2009), *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD Thesis.

Tamal K. Dey, Gang Li, Jian Sun. (2005), *Normal Estimation for Point Clouds: A Comparison Study for a Voronoi Based Method.* Eurographics Symposium on Point-Based Graphics, Ohio State Univ., Columbus, OH, USA pp.39–46.

J.G.Víctores; A.Jardón; M.F.Stoelen; S.Martínez; C.Balaguer, (2010) *ASIBOT Assistive Robot with Vision in a Domestic Environment*. Robocity2030 7th Workshop. Móstoles. Spain. Oct, 2010. Visión en Robótica. ISBN: 84-693-6777-3. Universidad Rey Juan Carlos. pp.61-74.