Proposal Team Pumas@Home SPL HSR 2018

Jesus Savage, Reynaldo Martell, Marco Negrete, Julio Cruz, Jesus Cruz, Jose Cruz, Jaime Marquez, Edgar Vazquez, Manuel Pano, Edgar Silva, Hugo Estrada, Hector Arce, Luis Alvarez, and Mauricio Matamoros

Bio-Robotics Laboratory, School of Engineering
National Autonomous University of Mexico
http://biorobotics.fi-p.unam.mx

Abstract. This paper describes the service robot Justina of team Pumas that has participated in the @Home category of the RoboCup and RoCKIn international competitions; as well as our latest applied research. These competitions had influenced our architecture in the development of better systems for our service robots by developing RGB-D representation of the environments; action planning using space state representation, and low or null texture objects recognition using RGB-D cameras. In our robotics architecture, the VIrtual and Real roBOt sysTem (VIRBOT), the operation of service robots is divided into several subsystems, each of them has a specific functionality that contributes to the final operation of the robot. By combining symbolic AI with digital signal processing techniques a good performance of a service robot is obtained. We consider that our robotics arquitecture, VIRBOT, can be transfered successfully into the Toyota Human Robot Support Robot (HSR).

1 Introduction

Service robots are hardware and software systems that assist humans to perform daily tasks in complex environments, to achieve this: they have to be able to understand spoken or gesture commands from humans; to be able to avoid static and dynamic obstacles while navigating in known and unknown environments; to be able to recognize and to manipulate objects and performing several other tasks that a person might request.

Our team has been participated in the category @Home continuously since the start of this competition at the RoboCup in Bremen in 2006. Our team obtained the third place in Atlanta in 2007, and has reached the finals in 2014 and 2015, this year, in the Robocup 2017, our team obtenied the fourth place and an award for the "Best in Speech Recognition and Natural Lanuguage Understanding".

The paper is organized as follows: section 2 enumerates the hardware and software components of our robot Justina; section 3 presents overview of the latest research developments in our laboratory; and finally, in section 4, the conclusions and future work are given.

2 Justina's Robotics Architecture

2.1 Hardware Configuration

Our service robot Justina, see figure 1, has the following hardware configuration:

ACTUATORS:

- **Mobile base:** Omnidirectional through differential pair configuration and omnidirectional wheels.
- **Manipulators:** 2 x 7-DOF anthropomorphic arms with 10 Dynamixel servomotors each.
- **Head:** 2-DOF (Pan and tilt) built with Dynamixel servomotors.
- **Torso:** 1-DOF (Elevation) through a worm screw and a configuration of gears.
- **Speakers:** Two speakers to generate synthetic speech.

SENSORS:

- **RGB-D Camera:** Microsoft's Kinect sensor
- **RGB Camera:** Logitech Pro C920 Full HD.
- **Microphone:** Rode NTG2 directional microphone.
- **Array of Microphones:** An array of four microphones to detect sound sources.
- **Laser:** Hokuyo rangefinder URG-04LX-UG0.

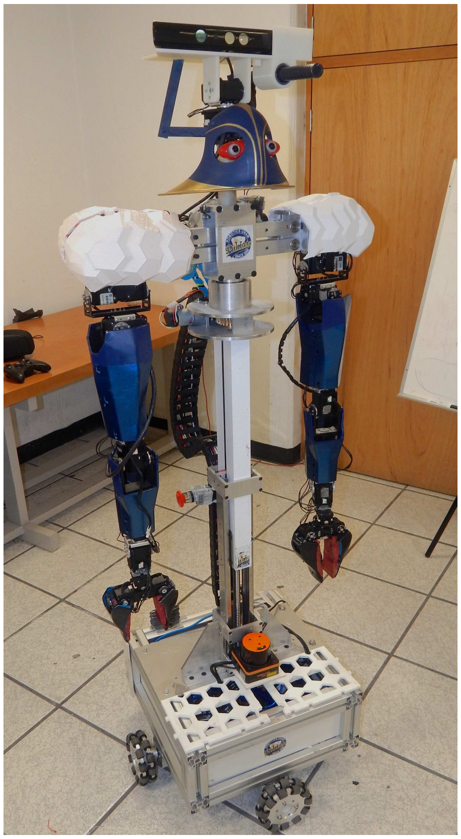


Fig. 1: Robot Justina

2.2 Software Configuration

Our software configuration is based on the VIRBOT architecture [1], which provides a platform for the design and development of software for general purpose service robots, see figure 2. The VIRBOT architecture is implemented in our robots through several modules that perform well defined tasks [2], with a high level of interaction between them. The principal framework used for interaction is ROS, where a module is represented by one or several ROS's nodes. Also, for modules using the Microsoft operating system, we use our own middleware called Blackboard to link them with ROS nodes running on Linux. In the following sections are explained each of the layers of the VIRBOT system.

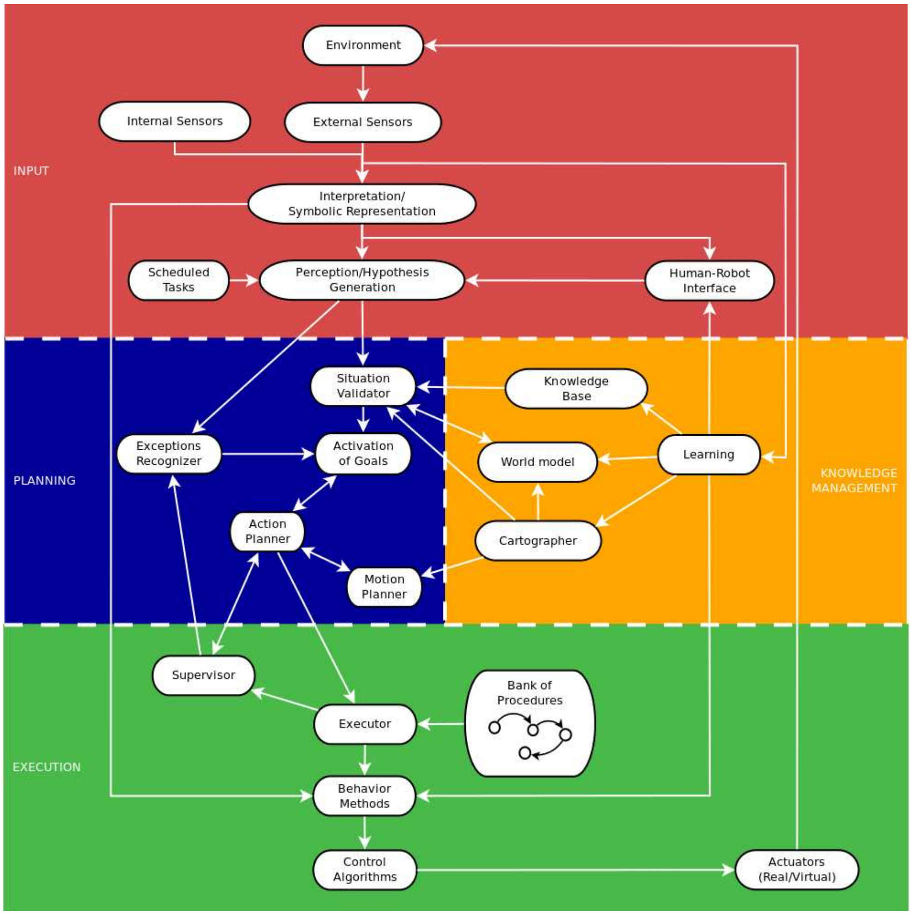


Fig. 2: Block diagram of the ViRBot architecture.

2.3 Inputs Layer

This layer process the data from the robot's internal and external sensors, they provide information of the internal state of the robot, as well as, the external world where the robot interacts. In some of Justina's designs it has lasers, sonars, infrared, microphones and stereo and RGB-D cameras. Digital signal processing techniques are applied to the data provided by the internal and external sensors to obtain a symbolic representation of the data, as well as, to recognize and to process voice and visual data. Pattern recognition techniques are used to create models of the objects and the persons that interact with the robot. With the symbolic representation this module generates a series of beliefs, that represent the state of the environment where the robot interacts.

2.4 Planning Layer

The beliefs generated by the perception module are validated by this layer, it uses the Knowledge Management layer to validate them, thus a situation recognition is created. Given a situation recognition, a set of goals are activated to solve it. Action planning finds a sequence of physical operations to achieve the activated goals.

2.5 Knowledge Management Layer

This layer has different types of maps for the representation of the environment, they are created using SLAM techniques. Also in this layer there is a localization system, that uses the Kalman filter, to estimate the robot's position and orientation. A rule based system, CLIPS, developed by NASA, is used to represent the robot's knowledge, in which each rule contains the encoded knowledge of an expert.

2.6 Execution Layer

This layer executes the actions and movements plans and it checks that they are executed accordingly. A set of hardwired procedures, represented by state machines, are used to partially solve specific problems, finding persons, object manipulation, etc. The action planner uses these bank of procedures and it joins some of them to generate a plan.

3 Current research

In this section is presented the current research developed in our laboratory to improve the performance of our service robots.

3.1 RGB-D representation of the environments

For the construction of roadmaps, 3D data is used to find the occupied and free space where a robot can navigate, it uses clustering techniques to find a representation of them. 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. 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 $q_j = (x_j, y_j, z_j)$, represents the spatial positions of the objects' points relative to the Kinect's position, mounted in top of the robot's mechatronic head.

The number of points captured by a 3D sensor in just one picture is immense, around 300,000 points, thus, it is necessary to compress the 3D data to a minimum, to be able to find a proper representation of them. For this, clustering techniques are used, vector quantization (VQ), which also they help 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.

Given a set of N_v vectors, $q_j = \{x_j, y_j, z_j\}; j = 1, ..., N_v$ that represent the position of the cloud points 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. A modified VQ algorithm for 3D [3] data 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. Figure 3 shows in the left side the original RGB image captured by the Kinect, in the center it is shown the free space clusters in green and the occupied space clusters, in purple, the black regions are those points with no depth information. In the right it is shown the resulting environment representation. Green dots represent the nodes, free space centroids, used to build the roadmap and calculate a path. Purple rectangles represent obstacles and the red lines are the path calculated by Dijkstra algorithm to reach the goal point, also colored in red.

To obtain real time performance the VQ algorithm was processed in parallel in a GPU programmed with CUDA, it was also implemented the whole process sequentially, using C++, to compare processing times and to test the effectiveness of the parallel implementation. Table 1 shows the results obtained comparing the VQ implementation in parallel and in sequentially of the environment's 3D data. As we can see for the results shown in this table, the parallel implementation of the VQ algorithm is in average 17 times faster than the sequentially one, thus this technique can be used by the robot in real time when it navigates.

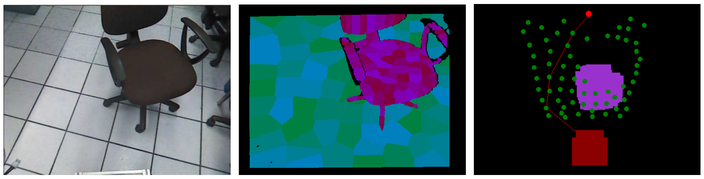


Fig. 3: **Left:** Original RGB image captured by the Kinect. **Center:** Free space clusters and occupied space clusters. **Right:** Resulting environment representation.

Sequential Time [s]	Parallel Time [s]
1.341	0.078
1.373	0.078
1.341	0.078
1.388	0.077
1.357	0.078

Table 1: Comparison of processing time for serial and parallel implementations of the VQ algorithm.

3.2 Action planning using space state representation

VIRBOT's task planning uses concepts from space-state search planning and hierarchical task networks, as the ones used in classical STRIPS-like planners.

The plan is generated by an inference engine, CLIPS, that it uses a set of rules that represent a hierarchical structure of tasks. The planning rules are useful for considering different situations, present in the environment, so that the robot can act accordingly. The mechanism to generate a new plan starts with a spoken command, the representation of it triggers a set of rules that generate the plan. The spoken commands representation is defined by performing a syntactical analysis and a semantic interpretation of them using a natural language technique called Conceptual Dependency [4]. The Robot is able to perform atomic operations like grasping an object, moving itself from on place to another, finding humans, etc. Then the objective of action planning is to find a sequence of this atomic operations to achieve the desired goal.

This research was evaluated under the General Purpose Service Robot (GPSR) test of the @Home category, and after generating 30 random commands of the set of commands of this test, the robot was able to accomplish, trough action planning, the required actions contained in the solution of the plans, shown in table 2.

3.3 Low or null texture objects recognition using RGB-D cameras

Currently, several robust techniques based on feature extraction and description exist for object recognition. However, if the objects are low textured, only a few

Type of actions	Percentage of success
Navigation	75
Object Recognition	59
Face Recognition	72
Object Handling	42
Answering questions	87

Table 2: Action planning success

number of features can be extracted, making the matching process unreliable. For these cases, we developed a method that combine three characteristics of the objects: color, size and shape, after a 3D detection and segmentation in a plane for each object.

Color information is extracted from the HSV space of the object's RGB pixels and it is represented by the histogram of the Hue components.

The size and shape is estimated from the object's point cloud, which are obtained using an oriented bounding box (OBB) of the points cloud. The shape is characterized using the Hu Moments [5] of the convex hull calculated from the points projected over the plane below them. Thus, with these representations an histogram is obtained for each of the objects and for the recognition process we compare the objects' histograms with the histogram of the object to be recognized.

The histograms are compared using histogram intersections, [6]:

$$H(I,M) = \frac{\sum_{j=1}^{n} \min(I_j, M_j)}{\sum_{j=1}^{n} M_j} \tag{1}$$

Where I is the histogram of the object to be recognized, M represents one of the stored objects' histograms and n is the number of bins in the histograms.

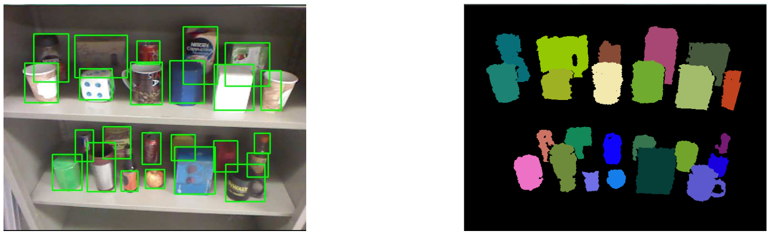
This method has been tested experimentally, showing fast and robust results for changes in light, scale, and rotation in a plane parallel to the plane below the object. Figure 4 shows an example of objects' recognition on a shelf (multiple planes).

Table 3 shows the results obtained by comparing this histogram method with a SIFT algorithm for 25 objects without or almost no texture.

System	% Recognized	% No recognized
Proposed	**91.333**	**7.666**
SIFT	22.333	77.666

Table 3: Comparison of recognition of objects using color, size and shape histograms and SIFT.

As we can see for the results on table 3 this technique outperforms the SIFT one.



(a) Objects can be segmented even with occlusions.

(b) Point clouds corresponding to each segmented object.

Fig. 4: Example of object segmentation on several planes.

4 Conclusions and future work

It is clear, that during the 11 years in which our team Pumas has been participated in the RoboCup and 2 years in the Rockin [7] in the category @Home, the performance and research developed, in the service robot area, in our laboratory has been improved considerably. Our service robot architecture, the VIRBOT, has been evolving according to the requirement that these robotics competitions asked each year. In these years, the full system has been improved, both in hardware and software, having reliable performance and showing promising results. Particularly, this year, we have a new omnidirectional mobile base for navigation and a new torso. In terms of software, we have change the way of conceiving the tests of the competition: from static state machines to inferred action planning generated by a rule based system. As for future work, the computer vision algorithms will be improved by using Hidden Markov Models (HMM) to have a better recognition of objects and persons. Also, it will be explored fault tolerant systems to help the robot to recover from failures. We consider that if our team becomes a parnert in the Joint Research for the standar Toyota Human Support Robot platform, the majority of our software, developed so far, can be used directly in this platform and our robotics research can be improved sustanciably.

References

1. *ViRbot: A System for the Operation of Mobile Robots*, Savage, Jesus and et al, RoboCup 2007: Robot Soccer World Cup XI, pp 512-519, Springer Berlin Heidelberg, 2007.
2. *The Design of Intelligent Agents: A Layered Approach*, Muller, Jorg P,Springer-Verlag New York, Inc.1997.
3. *Parallel implementation of roadmap construction for mobile robots using rgb-d cameras*, Marco Negrete, Jesús Savage, Jesús Cruz, and Jaime Márquez, OGRW2014, pages 184–187, 2014.
4. *Conceptual dependency and its descendants*, Steven L. Lytinen, Computers & Mathematics with Applications, 1992.

5. *Visual pattern recognition by moment invariants*, Ming-Kuei Hu, IRE Transactions on Information Theory, 8(2):179–187, 1962.
6. *Color indexing*, Michael J Swain and Dana H Ballard, International journal of computer vision, 7(1):11–32, 1991.
7. http://rockinrobotchallenge.eu/home.php