Pumas@Home 2016 Team Description Paper

Biorobotics Laboratory

National Autonomous University of Mexico, México DF 04510, MEX,
http://biorobotics.fi-p.unam.mx

Abstract. This robot is based on the ViRbot architecture, implemented by several modules that perform different tasks, through a cross-platform system called Blackboard together with the ROS platform. This year, our team focused on navigation in dynamic environments and task planning using different approaches, also the migration of several of our modules to the ROS platform.

1 Introduction

The service robots are hardware and software systems that can assist humans to perform daily tasks in complex environments. To achieve this, a service robot has to be capable of understanding commands from humans, avoiding static and dynamic obstacles while navigating in known and unknown environments, recognizing and manipulating objects and performing several other tasks that a person might request. This paper describes our service robot Justina and it is organized as follows:

In section 2 the ViRbot architecture is described as the platform for the robot software system. Section 3 enumerates the hardware and software components of robot Justina, and the functionality of each of them is described. Section 4 is an abstract of the latest research developed in our laboratory in order to improve the robot's performance and finally, in section 5, conclusions and future work are given.

2 Background: Platform and Architecture

This robot is based on the ViRbot architecture for autonomous mobile robot operation [1], which provides a platform for the design and development of software for general purpose service robots.

The ViRbot architecture defines a human-robot interaction interface made of three main layers (see figure 1):

- **Input layer:** Includes all algorithms related to the acquisition of data from the environment.
- **Planning and Knowledge:** This layer performs most of the AI algorithms.
- **Execution:** Includes low-level control and supervision.

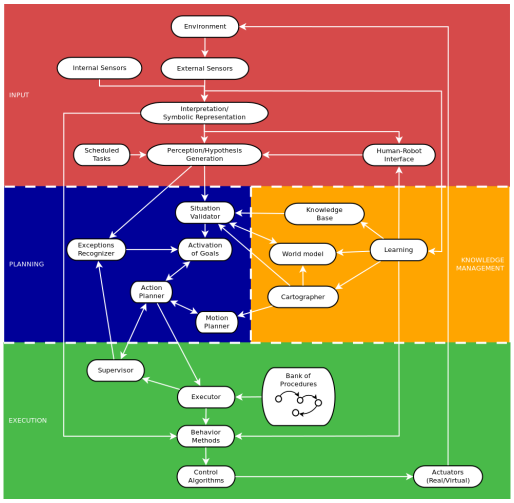


Fig. 1: Block diagram of the ViRBot architecture.

The implementation of ViRbot is made through several modules (executables) that perform well defined tasks and have a high interaction. The information exchange between modules is made through a central module called Blackboard (BB), which supports shared variables, with publisher/subscriber pattern, and message passing. Blackboard was developed with C# and the .NET framework, and there are currently APIs to build BB modules for languages C#, C++ and python. It runs on windows and linux-based systems (with the help of Mono, the open source implementation of Microsoft's .NET framework) such as Ubuntu and Raspbian (as command line interface). Also, a BB module has been developed for an Android system and a CLIPS interpreter has been used with the help of PyCLIPS, expanding the number of platforms and languages available to use with it.

BB's commands and shared variables have a certain equivalence to ROS's services and topics respectively. Integration between modules working with these platforms has been accomplished through the implementation of a Black-Board Bridge module, which communicates with both systems. Figure 2 shows a block diagram of how we interconnect Blackboard and ROS.

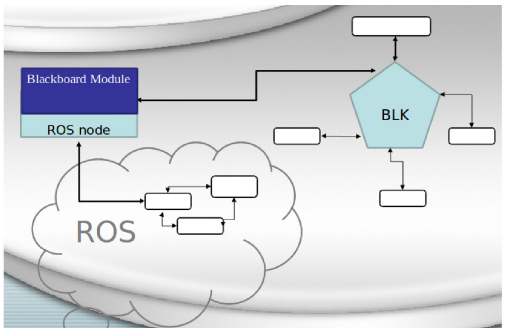


Fig. 2: Communication scheme between Blackboard and ROS.

3 Hardware and Software

Robot Justina has the following actuators:

– Differential mobile base (self design) with Arduino Mega for control and Pololu RoboClaw board as motor driver.

– Two arms with anthropomorphic design: 7 DOF implemented with 10 Dynamixel servomotors and CM-700 microcontroller board for control and path planning.
– Mechatronic Head with 2 DOF (Pan and tilt) built with Dynamixel servomotors.
– Speakers.

and the following sensors:

– One Kinect sensor and one RGB camera.
– Directional microphone.
– Laser rangefinder Hokuyo URG-04LX-UG0

Fig. 3: Robot Justina

3.1 Software

Robot Justina uses computers running both Linux and Windows, and modules programmed in C#, C++, Python and CLIPS [2]. Modules currently running on Justina are:

– Action Planner: Details are given in subsection 4.1.
– Simple Task Planner: It is a bank of procedures that involves simple and repetitive tasks easily achieved by state machines, e.g. grasping an object, searching for a face, aligning with an edge, etc.
– Motion Planner: Details are given in subsection 4.2.
– Object Finder: This module performs image and point clouds processing, object recognition using SIFT [3] + histogram comparison and geometric feature extraction of the environment (planes, lines, corners, spikes).
– Person Finder: Multiple face detection and recognition using VeriLook SDK (commercial software).
– Speech Recognition: Throws hypothesis (text strings) of recognized voice with a confidence ranking using Microsoft SAPI 5.3 [4] (commercial software).
– Speech Generator: Voice synthesizer using Microsoft SAPI 5.3 [4] with the Loquendo Susan voice (commercial software).

4 Current research

4.1 Planning using plan-space search and hierarchical task networks

The task planning is implemented as an expert system with logical programming. It adopts concepts from plan-space search planning and hierarchical task networks (HTN).

On the one hand, the task networks would be useful to specify dependencies between tasks that would be difficult to represent with a detailed description of the environment and the effects of actions on it, like the ones used in classical STRIPS-like planners. On the other hand, it uses the plan-space search in order to make planning as objective oriented as possible, so depending on the current situation and the currently active tasks, each task can decompose in one plan or another.

The plan specification is done through facts that represent a hierarchical structure of tasks, and each task can have several planning rules. The planning rules would be useful for considering different situations, present in the environment, so the robot can act accordingly.

A graphical tool for designing the plans was also developed. It uses the formalism of High Level Petri Nets (HLPN) to model the planning rules. Each rule includes one transition that separates its preconditions from its effects. In general, places in the Petri nets diagram would be labeled or named with a syntax that resembles first order logic predicates, and they would correspond with facts about the world or actions to be executed.

4.2 Navigation based on behaviors

The Navigation system is composed of several subsystems performing different tasks in different levels of abstraction: a task planner, a set of behaviors controlling the mobile base and another set controlling the mechatronic head, a localization subsystem, which contains the world representation and a Kalman Filter for estimating the robot's position, and a perception module, responsible for processing the raw sensor data. Figure 4 shows a block diagram of the different subsystems and its connections.

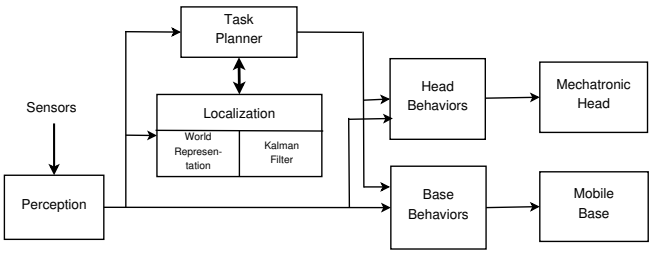


Fig. 4: The Navigation System

Justina's perception module includes object and face detection and recognition, natural landmark extraction, speech recognition, detection of obstacles and their position, skeleton detection and proprioception, that includes odometry and position estimations of the head and arms. The navigation system uses the subprocesses of odometry, landmark extraction and obstacle detection.

Landmarks are extracted from laser readings and point clouds generated by the Kinect sensor, using an algorithm based on the work of [5].

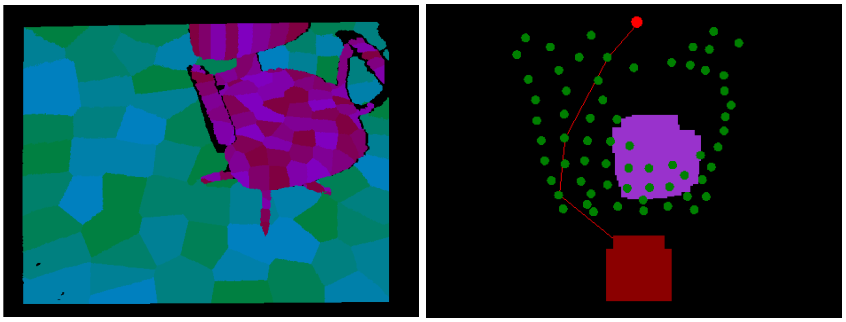
World Representation includes a geometric representation of the obstacles in the environment and a set of nodes used for path planning. Such path planning is made using the Dijkstra algorithm, considering information contained in the world representation and the perception module. Localization is made using a Kalman Filter.

The mobile base and the mechatronic head are controlled by a set of behaviors. Head is moved by two behaviors: the first one tries to point the head towards the nearest landmarks and the second one, towards the nearest obstacle. Mobile base is controlled by three behaviors. First behavior tries to move the robot towards a given goal point. The second one, avoids obstacles using potential fields. Laser readings are used for this purpose. The third behavior is checking if there are obstacles with which the robot could crash and stops the robot in case of collision risk.

The navigation system also builds roadmaps when it is moving to improve the obstacle avoidance. Roadmaps are constructed following the next steps:

- Point cloud acquisition from the kinect sensor.
- Transformation to the robot frame coordinates.
- Separation of free and occupied space (considering the z-coordinate of the points).
- Clusterization of free and occupied space by vector quantization techniques.

Centroids of free space are taken as nodes to calculate paths and clusters of occupied space are taken as obstacles. See figure 5. A more detailed description of the roadmap construction method is given in [6].



(a) Free space clusters are colored in green and occupied space clusters, in purple. The black regions are those points with no depth information.

(b) Resulting environment representation. Green dots are nodes, purple rectangles represent obstacles and the red lines are the calculated path.

Fig. 5: Roadmap construction.

4.3 Heterogeneous computing for point cloud processing

For a service robot, it is crucial to process a big amount of sensor's data in real-time to understand the environment around and react according to it. To achieve this, a device with high computing capabilities and concurrent processes running all the time is needed. Ideally one process for each sensor.

One of the most time consuming tasks in a service robot is the processing and analysis of 3D point clouds, where several thousand of readings per second can be obtained from the sensors.

For this reasons, the performance of different algorithms for point cloud processing has been tested using different heterogeneous computing systems. The systems tested are: CPU+GPU, CPU+FPGA, and CPU multicore. The tested algotihms are:

- Features extraction (normals calculation based on integral images [7])
- 3D planar surface segmentation (using RANSAC [8])

Figure 6 shows an example of the comparison of the execution time for the RANSAC algorithm using different systems. This algorithm is commonly used for plane segmentation in a 3D point set.

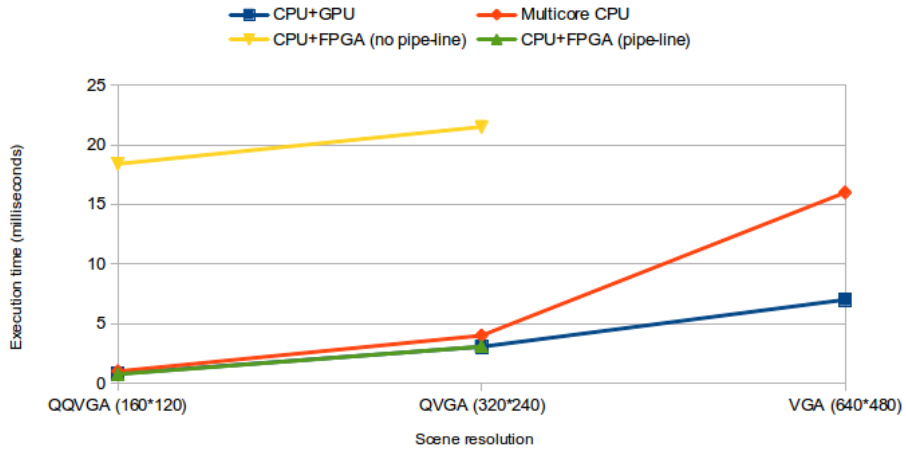


Fig. 6: Comparison of the RANSAC algorithm using different architectures.

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

Currently, several robust technics based on feature extraction and description exist for object recognition. However, if the objects are low textured, only a few number of features can be extracted, making the matching process unreliable. For these cases, we developed a method that combine three characteristics: color,

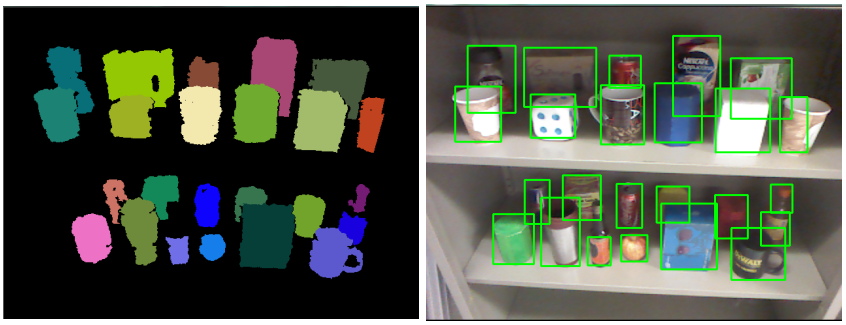
size and shape, combining the color and depth information for the recognition process, after a 3D detection and segmentation in a plane for each object.

Color Information is extracted from the HSV space of the object's pixels and it is represented by the histogram of the Hue component, but only for pixels with Saturation and Value above certain threshold. For pixels below these thresholds, two more bins are added to the histogram. For campaign histograms we used the histogram Intersection [9].

The size and shape is estimated from the object's point cloud, which are obtained using an oriented bounding box (OBB) of the point cloud as follow: the base of the OBB is obtained from the oriented bounding rectangle of the projection of the points in the plane below them. The heights are obtained from the maximum distance of the points to the plane. The shape is characterized using the Hu Moments [10] of the convex hull calculated from the points projected over the plane below them.

For the recognition process we compare in three steps: size, shape and color characteristics, removing candidates below a certain threshold for each step. At the end, from the remaining candidates, we select the best one according to a color-based similarity function.

These method has been tested experimentally in the Rocking robotics competition, where the team Pumas obtained a second place in the "object perception test", showing fast and robust results for changes in light , scale, and rotation in a plane parallel to the plane below the object. Figure 7 shows an example of object recognition on a shelf (multiple planes).



(a) Point clouds corresponding to each segmented object.

(b) Objects can be segmented even with occlusions.

Fig. 7: Example of object segmentation on several planes.

5 Conclusions and future work

The ViRbot system has been successfully tested in the Robocup@Home category since the Robocup competition of Atlanta 2007. The team has get into the finals

the last two years (2014 and 2015) in this competition. Also, in 2014, the team participated in the RoCKIn Competition wining a 1st place in "Catering for Granny Annie's Comfort" test and a 2nd place in the "Spech Understanding" test. In the 2015 edition of the same competition, the team won the second place in the "Object perception" test.

In these years, the full system has been improved having reliable performance and showing promising results. Also the Blackboard system grants us a versatile platform for developing subsystems, without the restriction of an operating system or a specific programming language.

As future work, algorithms for object and face recognition based on 3D features are being developed. Also, SLAM techniques are being tested using 3D information. For the manipulation task, a new control algorithm and a path planning system will be implemented for the arms module.

References

1. Jesus Savage and et al. Virbot: A system for the operation of mobile robots. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, volume 5001 of *Lecture Notes in Computer Science*, pages 512–519. Springer Berlin Heidelberg, 2008.
2. Gary Riley. *CLIPS Reference Manual Version 6.0. Technical Report Number JSC-25012.* Software Technology Branch, Lyndon B. Johnson Space Center, Houston, TX, 1994.
3. David G. Lowe. Distinctive image features from scale-invariant keypoints, 2003.
4. Microsoft. Speech server - microsoft corporation. `http://www.microsoft.com/speech/speech2007/default.mspx`, 01 2011.
5. R. Yan, J. Wu, W. Wang, S. Lim, J. Lee, and C. Han. Natural corners extraction algorithm in 2d unknown indoor environment with laser sensor. In *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, pages 983–987. IEEE, 2012.
6. Marco Negrete, Jesús Savage, Jesús Cruz, and Jaime Márquez. Parallel implementation of roadmap construction for mobile robots using rgb-d cameras. In *OGRW2014*, pages 184–187, 2014.
7. Dirk Holz, Stefan Holzer, RaduBogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In Thomas Röfer, N.Michael Mayer, Jesus Savage, and Uluc Saranlı, editors, *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 306–317. Springer Berlin Heidelberg, 2012.
8. Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.
9. Michael J Swain and Dana H Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.
10. Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.