Markovito Team Description Paper for RoboCup@Home 2020

L. Enrique Sucar[1] Eduardo F.Morales[1] Sergio A. Serrano[1]
Arquímides Méndez-Molina[1] Reinier Oves García[1]
David Carrillo-López[1] Esaú Escobar-Juárez[1]

February 4, 2020

Abstract. In this paper we present the current status of Markovito Jr., a service robot developed by the Markovito team at INAOE. Our robot is based on a RB1 robot platform and incorporates a set of general purpose modules that are integrated in a layered behavior-based architecture implemented on the Robot Operating System (ROS). The current research is focused on the development of novel algorithms for object detection and manipulation, people tracking, learning, as well as improvements on our modular software architecture. Important results were recently obtained by our group in robust indoor navigation and grasp confirmation at Mexican International Conference of Artificial Intelligence, 2019. In the context of robotics competitions our team has achieved important results in the national contests and also has previous participation at the International RoboCup@Home.

1 Introduction

Robotics in its evolution has been leaving aside the repetitive tasks of controlled industrial environments and, with a new generation of skilled robots in the execution of domestic tasks known as service robots, is getting involved in dynamic environments with human interaction. When introducing service robots into society, they are expected to have skillful mechanisms to perform most of the tasks a human can perform.

Service robotics is a challenging, and exciting field that for several years has motivated our research group Markovito [1] at INAOE. With our robotic platform we actively develop research in many subfields of service robotics. The Mexican Tournament of Robotics (TMR) has been the main competitive platform to test our results annually.

[1] Markovito Home Page: `http://robotic.inaoep.mx/~markovito/`

Besides the good historical results obtained in the robotics competitions, the philosophy of our group is mostly focused on research in the various areas related to service robotics. As a result, our group has achieved several publications in international journals and conferences, and has served for several students from INAOE to successfully develop their master's and doctoral theses.

In this Team Description Paper we present the current research carried out by our group, which is directly related to achieving the fundamental skills required by a service robot in the context of the RoboCup@Home contest. The rest of the paper is structured as follows: in section 2 we describe our team in more detail; section 3 presents the general software architecture we use; an overview of the latest research developments in our laboratory in the different robot skills is presented in section 4; finally the conclusions and some lines of future work are presented.

2 Group's description and research focus

Markovito's team is composed of researchers and students from the Robotics Laboratory of INAOE. The laboratory was created in 2007, however, the first research in the field of robotics dates back to 2001. Before our current robotic platform Markovito Jr. (2017 -), we used Sabina (2012 - 2017) and the original Markovito (2007 - 2012).

The team integrates graduate students and B.Sc. students on research stays. This year the team is composed by (3) researchers, (1) post-doctoral student, (3) Phd students and (3) students on research stays.

The Markovito team has participated in the RoboCup@Home category at previous RoboCup competitions in 2009, 2011, 2012 and 2016; in Turkey 2011 our team qualified for the second stage of the competition. In 2015 and 2016, the Markovito team won a 1st place in the Mexican RoboCup@Home competition.

Currently, our group's main research topics are computer vision, speech recognition, people tracking, safe indoor navigation within dynamic environments, task planning, robust object manipulation and gesture recognition. Some details of our current work are explained in section 4.

3 Markovito's Architecture

Markovito's software architecture has been designed and developed as a layered behavior-based [1] architecture that uses ROS [2] for communication (see Fig. 1). The architecture has three different levels:

1. **Decision level:** This is the highest level in the architecture, where a state machine is used as a global coordinator. A particular state machine is specified for each task to obtain a determined behavior.

2. **Execution level:** Modules in this level interact with the functional level through ROS architecture. This level includes the modules to perform basic tasks such as navigation, object manipulation, speech recognition, etc.

3. **Functional level:** At this level the modules interact with the robot's sensors and actuators, relaying commands to the motor or retrieving information from the sensors.

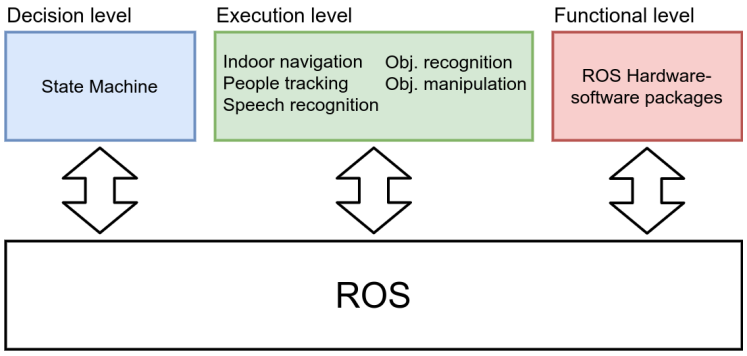

Fig. 1. Software architecture of Markovito: a layered behavior-based architecture that uses ROS for communication.

At the execution level, the robot's skills are encapsulated into general purpose modules. By doing so, programming the robot to solve different tasks becomes less time consuming and endows the architecture with scalability, which becomes more important as new modules are required.

4 Markovito's skills and current work

In this section we present the most recent work on Markovito's skill set, which encompasses indoor navigation in dynamic environments, people tracking, speech recognition, object recognition and robust object grasping and manipulation.

4.1 Indoor Navigation

The navigation algorithm that we are currently using (with some modifications) was designed by the Active Vision Group (AGAS) [3]. They present and explain the use of their software which they call Homer GUI, Homer mapping and Homer navigation corresponding to the graphical user interface, mapping algorithm and navigation algorithm, respectively. This software uses a particle filter to solve the SLAM problem and the A* algorithm for path planning.

The navigation system developed by AGAS was modified by integrating a depth image sensor, which gathered point clouds of the robot's surroundings and projected them onto the two dimensional plane (that corresponds to the floor) [4], as shown in Fig. 2. The projected points were combined with laser data so that the robot could identify obstacles at different heights above the floor, unlike the original system which could only detect obstacles at the level of the laser sensor (usually installed in the robot's mobile base). Additionally, our navigation system is able to re-plan a path to its target location when static (*e.g., furniture*) and dynamic obstacles (*e.g., people, pets*) are hindering, which is important for the safety of the robot and people in a real scenario.

(a) Laser contribution (b) RGB-D contribution (c) Fused data

Fig. 2. Static occupancy grid map used for Small Scattered Obstacles scenario integrating laser and RGB-D (figure taken from [4]).

4.2 People tracking

Our people tracking system is based on visual information obtained from the robot's RGB and depth image sensors. The system is able to keep track of multiple people within its range of view by performing at every instant of time a procedure that follows four main steps:

1. Segmentation on the depth image is performed to obtain a vector of blobs and their position with respect to the robot's framework.
2. In order to track the blobs, those from the current instant are matched to those from the previous one, based on the Euclidean distance.
3. The Single Shot Detection (SSD) neural network [5] is employed to recognize people in the RGB image of the current instant, if a person is found then its bounding box is extracted.
4. The bounding boxes of people found in the previous step are matched against the depth image in order to filter out those blobs that are not people. In Fig. 3 is shown an example of the robot locating a person within its map.

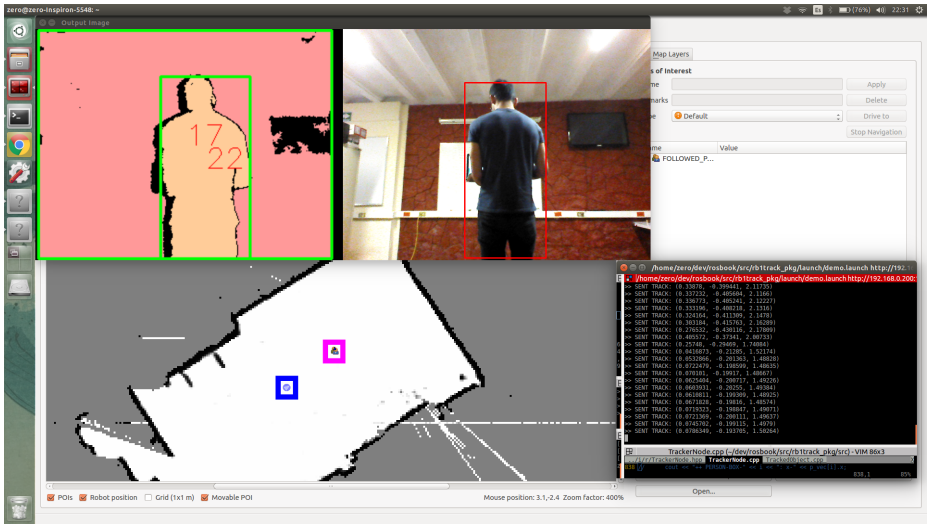

Fig. 3. Example of the tracking system locating a person within its map. The upper left image shows the depth image segmented into two blobs: a person (orange) and the background (pink). Next to the depth image, in the RGB image the person's bounding box is extracted using a pre-trained CNN. In the bottom of the figure the location of the robot (blue square) and the tracked person (fuchsia square) are shown within the robot's map.

In order to follow somebody, after the system projects the position of the person of interest in the map the robot uses to navigate, the robot knows where the person is within the map and can safely get closer without crashing into potential obstacles, for instance, tables and chairs. Additionally, because of the simplicity of our approach, it does not require of a GPU to work in real time.

4.3 Speech Recognition and Synthesis

We use the Pocket Sphinx [6] engine for speech recognition. Several grammar files (or sets of recognizable phrases) are defined depending on the task to be performed by the robot. The system can only identify the phrases or words defined in its grammar. The coordinator sets the right set of phrases to be used by the speech recognition module on each task. For synthesis we use Festival [7].

4.4 Object and people recognition

Among the sensors Markovito is equipped with, imagery (both RGB and depth) provide the most descriptive information about the robot's immediate surroundings. However, in order to exploit such rich data and extract from it useful information, we employ convolutional neural networks (CNN). Currently, our robot identifies two main categories of entities in images:

1. **People:** For tasks that require Markovito to identify people's location within the room, it uses the SSD network [5], which returns a list of boxes that enclose people in the RGB image. Moreover, our robot is also able to perform face recognition, based on the tiny face detector network [8] which is capable of identifying faces even with low resolution. Similar to SSD, the tiny face detector returns boxes that enclose the faces found in an image.
2. **Objects:** Unlike for people recognition, in which we use a CNN to crop areas of an image that contains a person or its face, for object recognition our system performs segmentation on depth images to locate where objects are with respect to the robot's framework. Currently an important constraint is that the objects must be over a flat surface. Then, it extracts the bounding boxes in the RGB image that correspond to the location of the objects in the depth image. Finally, the RGB snippets are classified with a pre-trained CNN (VGG [9]) that has been retrained to classify a set of known objects. Currently, one research line of the group is focused on the creation of an incremental learning scheme of objects in the scene (not necessarily on flat surfaces) so that a human instructor can progressively teach new objects to the robot. Once the robot incorporates the new objects learned, the idea is that it will be able to detect any appearance of these in the scene simultaneously. We are currently reviewing YOLO[10] and SSD[5] approaches to include some modifications to add new objects to the database to make the retraining as inexpensive as possible.

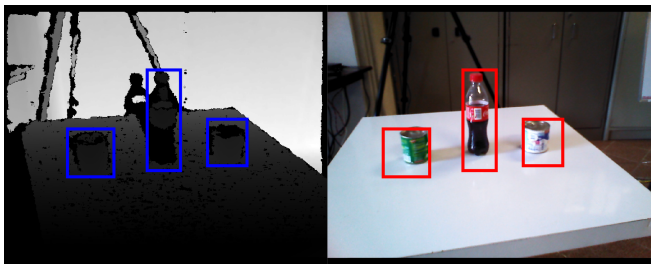

Fig. 4. Objects are located by performing segmentation on the depth image.

4.5 Object manipulation and grasping

For tasks that require the robot to pick up and move objects, we employ *GraspIt!* [11] in tandem with *MoveIt* [12], which are two libraries developed for object manipulation tasks. The former is for identifying the best grasp position for a particular object and the robot's end effector, while the latter performs motion planning.

Although *GraspIt!* and *MoveIt* have shown to be quite effective and accurate, a robot remains prone to fail at grasping objects as a consequence of exogenous factors, *e.g.*, somebody moving the target object after the manipulator started approaching, or the object slips from the robot's gripper. Moreover, since in many cases other sub-tasks will depend on the robot's success in grasping an object (as part of a larger task), failing at picking up an object might have a high cost, therefore, feedback is required.

In order to endow the robot with a source of feedback, we developed a grasping confirmation mechanism that is low time consuming and quite robust [13]. We specified a position for the robot's gripper in which it could be observed by the robot without occlusion (as shown in Fig. 5). Fine tuning was performed to a pre-trained neural network by training the fully connected layers to identify two classes of images: i) empty gripper and ii) gripper holding something. Thus, after our robot attempts to grasp an object, it takes the gripper to the predefined position and verifies if it succeeded, if not, it can decide whether to try it again or move onto the next sub-task.

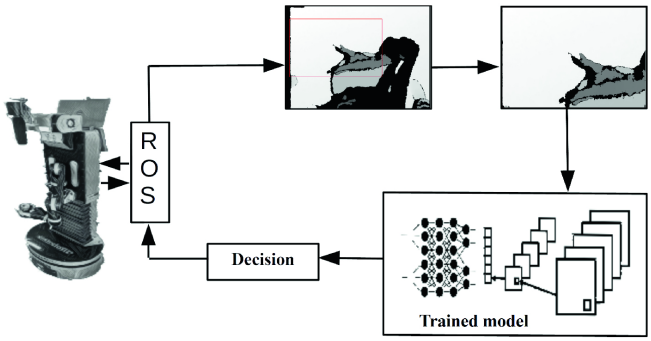

Fig. 5. Flowchart of our grasp confirmation mechanism (image taken from [13]).

5 Conclusions and future work

In this document, Markovito's software architecture has been described. We have developed a set of general purpose modules, integrated in a layered behavior-based architecture that communicates its modules with ROS. These features enable Markovito to perform on tasks that are made of sub-tasks that can be solved with Markovito's skills, such as the RoboCup@Home tasks. Based on this framework and different platforms, we have participated in the Mexican Robotic Tournament since 2007 achieving top positions each year. For instance, in 2015 and 2016 we won a 1st place in the Mexican RoboCup@home competition. We have also participated in the international Robocup@Home competition in 2009,

2011, 2012 and 2016. In order to suffice the needs of a real world scenario, we believe that integrating reactive behavior becomes a must, so that a robot is ready to solve tasks at any time. Also, as future work we would like to enable the system to dynamically build state machines based on task requests.

References

1. Ronald C Arkin, Ronald C Arkin, et al. *Behavior-based robotics*. MIT press, 1998.
2. Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.
3. Viktor Seib, Raphael Memmesheimer, and Dietrich Paulus. A ros-based system for an autonomous service robot. In *Robot Operating System (ROS)*, pages 215–252. Springer, 2016.
4. Orlando Lara-Guzmán, Sergio A Serrano, David Carrillo-López, and L Enrique Sucar. Rgb-d camera and 2d laser integration for robot navigation in dynamic environments. In *Mexican International Conference on Artificial Intelligence*, pages 661–674. Springer, 2019.
5. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
6. David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alexander I Rudnicky. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE, 2006.
7. Alan Black, Paul Taylor, Richard Caley, Rob Clark, Korin Richmond, Simon King, Volker Strom, and Heiga Zen. The festival speech synthesis system, version 1.4.2. *Unpublished document available via http://www.cstr.ed.ac.uk/projects/festival.html*, 2001.
8. Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 951–959, 2017.
9. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
10. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
11. Andrew T Miller and Peter K Allen. Graspit! a versatile simulator for robotic grasping. 2004.
12. Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.
13. Sebastián Salazar-Colores, Arquímides Méndez-Molina, David Carrillo-López, Esaú Escobar-Juárez, Eduardo F Morales, and L Enrique Sucar. A fast and robust deep learning approach for hand object grasping confirmation. In *Mexican International Conference on Artificial Intelligence*, pages 601–612. Springer, 2019.

Markovito Software and External Devices

The base platform is a Robotnik RB-1 model, modified whith a series of aftermarket devices to improve the robot's sensing capabilities.

Robot's Software Description

For our robot we are using the following software:

- Platform: Ubuntu 14.0 Operating System.
- Speech recognition: Pocket Sphinx.
- Face recognition: Tiny Face Detector.
- Object recognition: custom software based on CNN classification over depth images segmentation.
- Arm control: Kinova driver, MoveIt.
- Manipulation: GraspIt.
- Grasp Confirmation: custom software based on CNN classification over depth images segmentation.
- Obstacle avoidance: custom modification of the AGAS navigation package Homer.

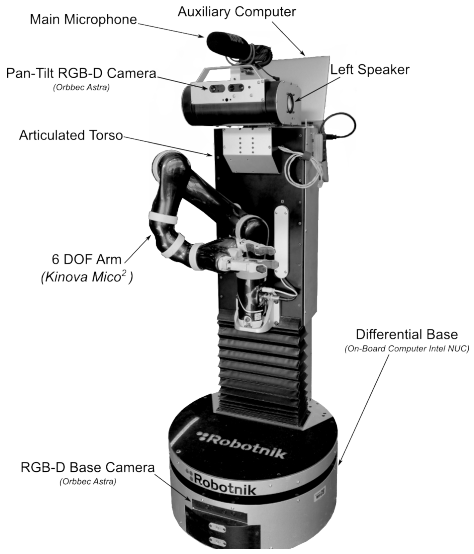

Fig. 6. Robot Markovito

External Devices

Markovito robot relies on the following external hardware:

- Torso attached Intel Laptop.
- Wireless accessed GPU Laptop.