Walking Machine @Home
2018 Team Description Paper

Jeffrey Cousineau and Philippe La Madeleine

École de Technologie Supérieure
1100 rue Notre-Dame Ouest, Montreal, QC, Canada H3C 1K3
http://walkingmachine.ca, walking@ens.etsmtl.ca,
https://github.com/WalkingMachine

Abstract. This paper gives details about the RoboCup@Home league team Walking Machine, from ETS University in Montreal, Canada for the next competition in his hometown, Montreal, in July 2018. The robot from Walking Machine, named SARA for "Système d'Assistance Robotique Autonome" (in English, Automated Robotic Assistance System), is a robot entirely built by the scientific club from ETS, mainly composed of undergraduates students. The robot is used for social interaction with humans, navigation and object manipulation. This document shows the electrical, mechanical and software properties and functionalities of SARA. It specifically emphasizes on human following, object and people recognition as well as navigation, manipulation and human-robot interaction.

1 Introduction

Walking Machine's team is a young team from Montreal, Quebec, in Canada, composed of engineering student in the field of mechanical, electrical and software engineering. We have been working on our robot for the last year in prevision of the Robocup at Home competition. As this would be our third participation, we learned a lot on our second attempt last year and have made various modifications to get better results, mostly on the software side. In the past, the team went in many competitions like the Eurobot, but made the leap for the RoboCup@Home competition to get a bigger challenge.

SARA, our creation, was designed for polyvalent human-robot interaction as well as efficient navigation and object manipulation. Our robot is mounted on four mecanum wheels powered by Roboteq drives, has one arm mimicking a normal human arm, and sensors for communication and navigation. Our team has developed knowledge in object and people detection/recognition, as well as navigation using a laser scanner, odometry on the wheels and a Asus Xtion camera. All of these parts are interfaced through ROS (Robot Operating System).

2 Electrical and mechanical design of SARA

2.1 Electrical

As an improvement this year on the electrical side of our robot, we put a lot of efforts in the organization of the electrical systems, we made it much clearer and safer. We put a lot of protection for every sub-system since it's easier to just change a fuse than rebuilding a whole electronic circuit board.

Another big change for us was also the new battery system. We went from a custom LiPo battery to a system using drill batteries. This change bring way more positive aspects than we tought. First of all, this change was mainly decide just before leaving for Japan this year. This gave ves us the opportunity to ship the robot without any batteries and to bring the batteries with ourself.

Second impact is that this type of battery is way safer than what we had before. We are using the drill charging station, that way we don't have to worry about overcharging our batteries. We have two batteries onboard, this way we have more than enough power. We also 3d printed the original connector so we can easily interchange the batteries with charged ones. Finally, because there's two batteries onboard, the robot can still run on one while we change the other one.

2.2 Mechanical

Some mechanical improvement were also made this year. There's mainly two things we improved on our robot, the arm and the base.

First of all, last year, our arm was made of 5 degrees of freedom, which makes some path plan a little more complicated and sometimes, impossible. Because of that, we decide to add 2 degrees of freedom with 2 dynamixel servo motors. Since then, we have much better results with our inverse kinematic and this also extend our range to get objects that are further.

Next thing we improve was the compactness of our robot base. On our first competition in Germany, we observe that our robot was too large, which caused some problem with our path planning around the doors. By relying on these observations, we decided to reduce the width which improved a lot the navigation capabilities.

3 Software

3.1 High-level task planning

For our task planning, we use a state machine software developed by team Vigir, one of the participant of the Darpa Robotics Challenge. This software is named FlexBe, for Flexible Behavior, is a block based interface for making state machines.

But we do not simply use FlexBe as is. We still need write our own blocks for it using its python API. To simplify our work, we started by splitting all of the Robocup@home scenarios into basic actions. We then identified all of these basic actions our platform could accomplish and we confined them in blocks named States, e.g. MoveArm, MoveHead etc. These blocks can then be assembled together to form a higher level of blocs we named Actions e.g. Pick, LookAt etc. Those Actions can then again be assembled into what we call ActionWrappers. The role of ActionWrapper is to allow interfacing our Actions with our natural language processing software(see natural language processing below). They receive the segmented text and translate it to computer understandable parameters using our Wonderland (see environment reasoning below) knowledge base and other sources of informations. This recursive structure allow us allow us to quickly develop our robot's behaviours.

3.2 Natural language processing

To analyse the detected speech, we rely on a software develop by the Semantic analytic group from the University of Roma and the Laboratory of Cognitive Cooperating Robots at Sapienza University of Rome. This speech analyzer, named LU4R for "Language Understanding For Robots", is composed of a server developed in Java which take as an input the detected sentence and the semantic environment surrounding the robot.

This server communicate through a REST service which give it the possibility to be compliant with all kind of platform. All you have to do is to launch the server locally which is compiled through a .jar file. Again, this give the opportunity to use this software on every platform, whether you're on Windows, Linux or Mac.

This software gives you the possibility to get different output representation. We choose the amr representation since it was the easiest one to understand and to implement.

We decided to build our own ROS wrapper, lu4r_ros to better interface it with our task planning approach. We first translate the response given by LU4R into simple format we call ActionForms. The ActionForms contains an action followed by all of its possible parameters as identified in the FrameNet Index of Lexical Units The ActionWrappers are then fed into a FIFO based priority manager and finally send to our task planner.

What is also interesting about LU4R, it's that it will use the semantic mapping in it's analysing process. All you have to do is to provide the correct pose for every object in the robot's environment. You can also precise various synonym for every object to get a better understanding of inputted sentence.

3.3 Object recognition

As a beginning team, we are still exploring various solution around the object recognition problem. Our first plan was to use the object recognition kitchen package from Willow Garage. But after using it in competition, we realised that the performance and the easiness to use wasn't what the approach we were looking for. As an alternative, we start using the YOLO (You Only Look Once) ros package (link).

YOLO is a real-time object detection. It does not only detect various object but it also predict the bounding boxes of the detected object. It use a single neural network which is applied to the image. Multiple regions are then created and are used to predict the bounding boxes. Each bounding boxes also contain are predicted probability which are used to filter the predicted objects. The advantage of this system is that it can detect multiple objects in a real-time scenario.

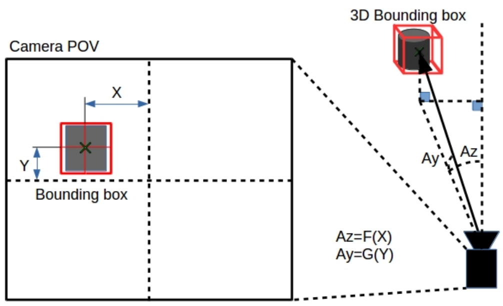

Fig. 1. 2D bounding box to 3D grasping pose, current technique

We use the ROS package to make our job easier since this gave us the possibility to directly get the recognized objects output into a ROS topic. We can also get the bounding box for each detected object. First step we had to do was to transform those 2D bounding boxes in 3D to get a specific pose according to our robot.

For the moment, we created the package wm_frame_to_box to approximate the object pose by the depth point of the center of the bounding box. Even though it can have flaws, this technique has also proven to be largely sufficient for most of our applications and most importantly, it uses way less processing power than the full 3D pattern matching we used before. This allow us to do real time object positioning, a capability we are proud of.

Afterward, to get better results and a better pose, we plan to substract the point cloud according to the bounding boxes. We will then use it with the point-cloud segmentation from the PCL library to extract the specific object and send it to a grasp identification package like haf_grasping. This technique, compare to what we are actually using would give us the possibility to grab a much wider variety of object.

Fig. 2. 2D bounding box to 3D grasping pose, future technique

For the moment, we are using the YOLO model but, we are looking forward to train our own dataset, just like we would do in competition. As an undergrad team, this is new for us but we have all the tools we need to overcome this.

3.4 Navigation

In addition of last year navigation stack, this year we are using the pointcloud-to-lasercan package as a lighter solution for 3D navigation. This ensure a safe navigation around object like table or chair, since only the legs are detected by the lidar laser. Another great thing with this implementation is the fact that we can set the maximum height for collision avoidance. That way, if there's an obstacle at head level, our robot will avoid it the same way it would avoir an object on the floor.

[image pointcloud_to_laserscan avec exemple d'une table]

We still use octomap as part of the collision avoidance system for the moveit arm planning to ensure there's no collision while moving to an object on top of a table.

3.5 Environmental reasoning

As novelty this year, we implemented our own solution for an evironment representation in a way that we think is simple and easy for everyone.

Our package named wonderland is an agnostic system in the same way than LU4R. It's composed of a server that received HTTP query based on a custom API.

The first thing that is needed when you start using this platform is to populate the database. This can be done manually if you already know the object position, but this can be done by your robot trough POST query. It's also possible to specify some room with a specific position relate to it. Once this is done, you can call our API and for example, juste by doing a GET request on the url http://wonderland:8000/object, this will return you the list of every object the database contains. It's also possible possible to filter the request by giving a known color link to the object or a room as a parameter.

Since the database is host as a server, you can access to it from everywhere. You could decide to export it to another computer, run in cloud or just put it on the robot system itself. It also give the possibility to have a dynamic knowledge, meaning that the robot can update it's knowledge of his environment in real-time.

4 Conclusions and future work

As you can see, since last year, we made large progress, starting with the improvement of our object detection module. Also, the implementation of a semantic representation of the environment is a big step for us since it helps our robot to have a better understanding of his surrounding. Finally, the various mechanicals modifications will help for a better navigation and a better management of our robot. So SARA is now a completely autonomous system able of interaction with the human as well as navigation in her environment. Using his multiple sensors, like laser sensor, depth camera, IMU and odometry our robot can analyse his environment, recognize people and objects and navigate through obstacles. It can then interact with objects and people using his voice recognition and voice synthesis abilities to keep track of his state. All of the requirements for the Robocup@home are met by SARA and it's a fast-evolving robot considering the team is really young and composed of undergraduates in various engineering programs. As future works, we will add a second arm to give us the possibility for two-hands manipulation. Also, we'll add a vertical translation to our body to give our robot a larger range for objects manipulation.

Robot SARA Hardware Description

Specifications for robot SARA are as follows:

SARA	
Base	Custom base with fully holonomic platform
Right arm	7 DoF custom arm made of Kinova motors
Neck	Tilt and pan unit using two Dynamixel MX-64R servo actuator
Head	Custom head made of RGB neopixels leds and Asus Xtion Pro
Gripper	Robotiq 2 fingers 140mm
Dimensions	Base : 0,61m. X 0,77m. Height : 1,68m.
Weight	~80kg
Additional sensors	Hokuyo UTM-30LX on base
Microphone	Rode microphone
Batteries	2x 20V Dewalt drill battery 5aH
Computer	1x Lenovo p50 with 32GB RAM and nVidia Quadro M2000 4GB, 1x Raspberry Pi 3

Table 1. Robot's hardware description

Robot's Software Description

For our robot we are using the following software:

- Platform: Robotic Operating System (ROS) Kinetic on Ubuntu 16.04
- Navigation, localization and mapping: Gmapping, AMCL, pointcloud_to_laserscan
- Face recognition: People
- Speech recognition: Google Speech API
- Speech comprehension : LU4R
- Speech generation: Svoxpico
- Object recognition: Darknet with YOLO v2
- Arms control: MoveIt and Kinova API
- Task executors: Flexbe
- World reprensentation: Wonderland

Fig. 3. Robot SARA

Team members

Jeffrey Cousineau, Philippe La Madeleine, Maxime St-Pierre, Jimmy Poirier, Philippe La Madelaine, Samuel Otis, Redouane Laref, Louis-Charle Labarre, Lucas Maurice, Léonore Jean-François, Nicolas Nadeau

References

1. LU4R Project - adaptive spoken Language Understanding For Robots.
2. Object Recognition Kitchen — object_recognition_core. `http://wg-perception.github.io/object_recognition_core/`.
3. pocketsphinx - ROS Wiki. `http://wiki.ros.org/pocketsphinx`.
4. robot_pose_ekf - ROS Wiki. `http://wiki.ros.org/robot_pose_ekf`.
5. Svox package : Ubuntu. `https://launchpad.net/ubuntu/precise/+source/svox/`.
6. J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. of the ASME. Journal of Applied Mechanics*, 22:215–221, 1955.
7. F.F. Sales. *SLAM and Localization of People with a Mobile Robot using a RGB-D Sensor:*. Master of Science Dissertation. University of Coimbra, 2014.
8. Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2016.
9. Stephen J. Wright. Coordinate descent algorithms. *Math. Program.*, 151(1):3–34, June 2015.