Università degli Studi di Genova

Software Architecture Project

RGB based object recognition - Module 3

Students: Mario Ciranni, Filippo Gandolfi
Professor: Fulvio Mastrogiovanni
Supervisor: Syed Yusha Kareem, Anthony Thomas
Year: 2018 - 2019

Contents

1 Overall goal of the project

The overall project is tilted "Constructing and navigating through a topological SLAM map". The objective is to develop a software architecture for a social mobile robot to achieve the desired objective.
The platform exploited for the project it has been the ROS-supported mobile robot MiRo, engineered by the company Consequential Robotics (fig.1). In particular, MiRo-b is defined as a fully programmable autonomous robot for researchers, educators, developers and healthcare professionals. It has six senses, eight degrees of freedom, an innovative brain-inspired operating system and a simulation software package [1].

Figure 1: MiRo by Consequential Robots

MiRO has three major developed areas: sensors, inspired by six different animals, actuators, fundamental for movement control, and the processing and communication tools, main center for the control of the robot (fig.2)[1]. All this components help MiRo to come alive.
In order to achieve the goal of the project MiRo is expected to mainly fulfill 2 tasks

1. autonomously explore and navigate the world
2. merge objects information provided by multiple perception systems.

[1]source: http://consequentialrobotics.com/miro-beta.

To achieve the latter task, the system should process information retrieved and gathered by different perception modules, namely

- Monocular SLAM for depth perception;
- RGB object recognition;
- Bluetooth beacons based distance estimation.

From within the image (fig.2)[2] can be highlighted the perception system that we have leveraged for this project. In particular,

- the Bluetooth module,
- the 2 HD 720p cameras located in the eyes (stereo vision),
- and the ultrasonic proximity and infrared detection sensors.

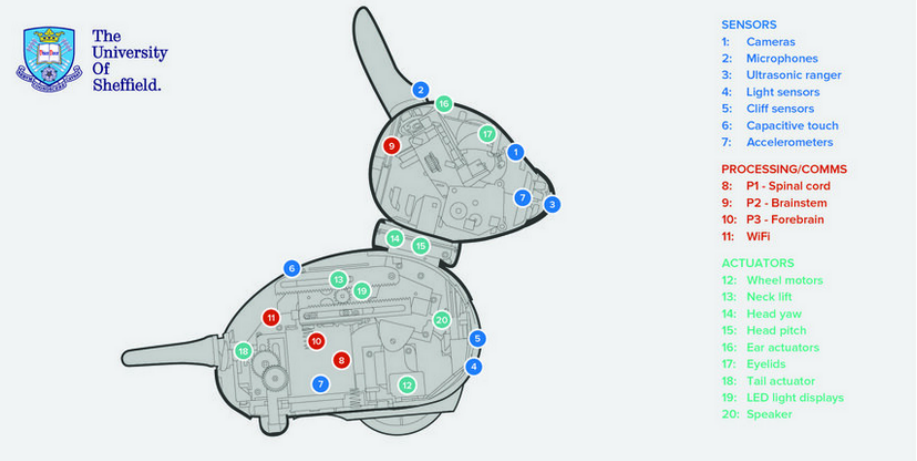

Figure 2: Techincal specifications of MiRo.

Eventually, the idea is to perform sensor fusion in order to integrate the information collected from the different modules in order to enrich the knowledge base and create a topological map as well as to create a user-friendly GUI to enable the user to control the robot.

[2]source: http://consequentialrobotics.com/miro-beta.

2 System's Architecture

2.1 Overall Architecture

Endowing the robot with the necessary capabilities to fulfill the aforementioned purpose, entailed devising different software modules, each one with a specific intent.
The modules are 5 and are divided as follows:

- **Module 1.** Navigation: MiRo navigating through the environment with the help of a bug algorithm based on computer vision.
- **Module 2.** Monocular SLAM: Acquire depth information from monocular vision of the moving mobile robot.
- **Module 3.** RGB based object recognition: Object recognition using machine learning tools.
- **Module 4.** Bluetooth beacons based distance estimation.
- **Module 5.** Interface between user and fused perception information: GUI (Graphic User Interface).

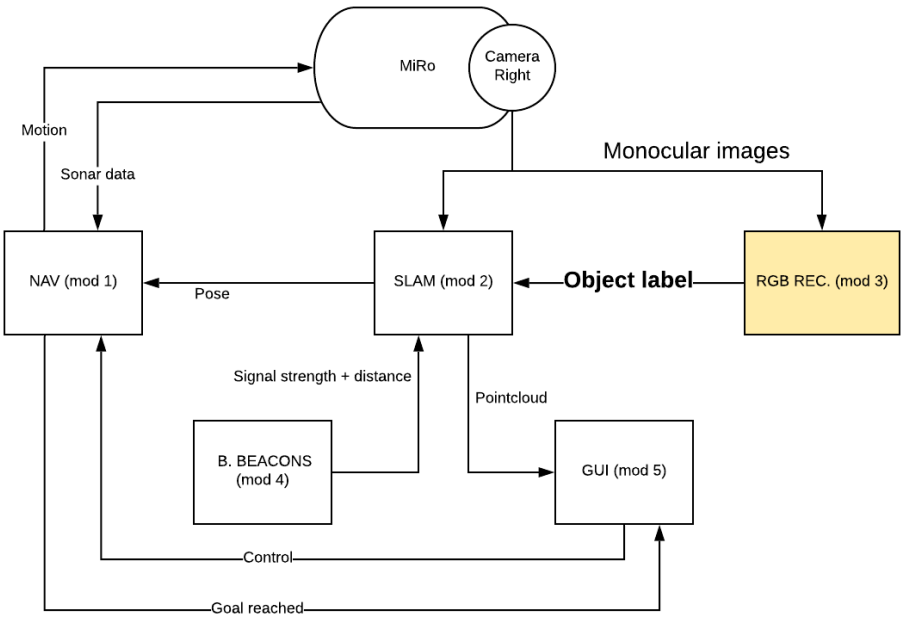

Figure 3: MiRo Project overall scheme.

Possibly the SLAM module represents the core one which is in charge of feeding both the Navigation module as well as the GUI in order to actually let the robot move through the environment and, as input, it retrieves from the RGB object recognition module the information about objects' labels in the

environment.
The module relies on a Monocular SLAM based depth perception. In particular, it has been chosen to use the ORB-SLAM2 [2] approach, which is a complete open-source simultaneous localization and mapping (SLAM) system for monocular, stereo and RGB-D cameras.
The input of the SLAM module are:

- the visual information retrieved from the right camera of MiRo,
- the object label retrieved from the object recognition module (module 3),
- the distance estimation retrieved from the Received signal strength indication (RSSI), from module 5 of the labeled Estimote beacons (see fig.4).

Figure 4: Estimote beacons

As output the SLAM module is able to retrieve the pose of the robot as well as the pointcloud in real-time, that are transmitted to the navigation module (module 1).
The navigation module, in turn, is in charge of actually steering the robot to a desired target, works essentially in 2 different modalities:

- autonomous
- manual.

In the case of the former, whenever the user from the Graphic User Interface (GUI) sets a desired goal position for the robot, MiRo tries to reach the goal position and at the same time, if an obstacle is detected from the sonar sensor, it tries to avoid it. Basically, whenever an obstacle is detected MiRo steers of an angle of 90° and moves forwards so to avoid the barrier; it keeps on doing it iteratively until the obstacle is avoided.
In the latter case instead the user by means of a Ps4 joystick is able to manually steer the robot to the desired target position.

2.2 The Module 3: RGB based object recognition

2.2.1 Description

The main goal of Module 3 is to enable MiRo to detect and recognize objects and to retrieve their pose (i.e. bounding boxes) as well as their names (i.e. labels). For the purpose a prettrained model has been exploited in the context of the Tensorflow framework and, in particular, the Tensorflow-Object Detection API is the main library on which the architecture is built on.
The model we have exploited is the SSD-mobilenet trained on the COCO dataset [3].
The final output of the module is simply a string label, e.g. chair, door, table, etc. Since the output of module 3 is the input of module 2, and we wanted to keep the information retrieved as simple as possible, we have set a threshold in order to validate the label retrieved for the object according to its score value.
The type of communication chosen it has been the Publish / Subscribe protocol on the topic `/adapted_message`.
We have made this choice since it was enabling an asynchronous communications and the sending node did not need to have an answer or a confirmation that the message had been well received from the others'.
When the object is detected and classified, this information is advertised on the topic and can then be retrieved from module 2 which, in turn, updates the map assigning the label to the target object. Also, the objects detected might come in handy for module 5 so to better discriminate on where Miro has to go according to the information retrieved and the willing of the user.

2.2.2 Architecture

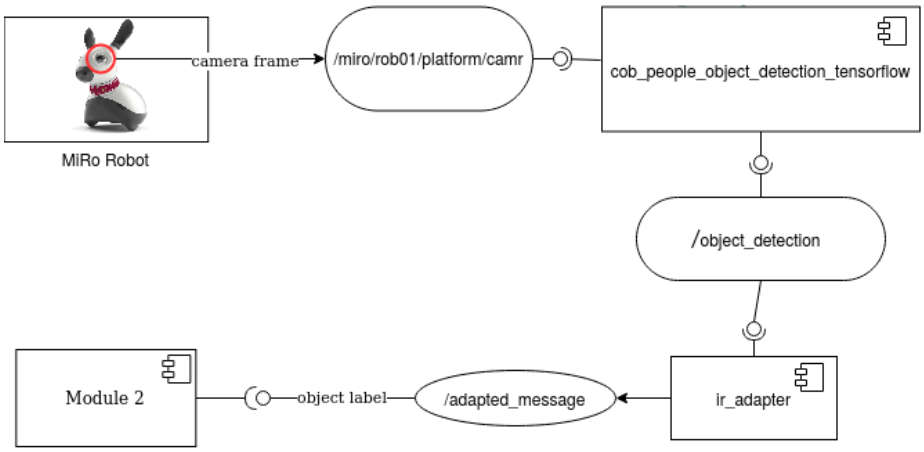

Figure 5: Module 3 Operational Diagram

There do exist substantially 2 ROS nodes within our software architecture:

- `cob_people_object_detection_tensorflow` and

- `ir_adapter`.

These nodes are 2 python scripts which are in charge of first recognising the image and then to filter the obtained output in order to have a simpler gathered information as input for the other modules, according to the different necessities. Changing the parameters on a .yaml file, which in turn, upon the launching of the architecture, gets loaded on the parameter server, we are able to decide wether to acquire the information from a RGB camera topic or directly from a webcam device.
Since MiRo publishes the information retrieved from the right camera to the ROS topic `/miro/rob01/platform/camr`, we have set this topic as input for the object detector.

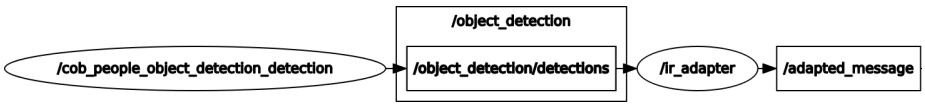

Figure 6: rqt_graph of the nodes and topics of our software architecture.

In the figure above is depicted the communication layer between the nodes within the architecture in the ROS framework (rqt_graph). The oval leaves are meant to represent nodes whilst the rectangular ones are topics.
`cob_people_object_detection_tensorflow.py` is the core node of the project. This scipt uses the *Numpy* and *Tensorflow* libraries. Numpy is a library that allow us to convert the stream of images in n-dimensional arrays. It is used both for input and output, where it send back the images with bounding boxes. The node exploits the cv_bridge functionality in order to take the advantages of using the OpenCV (see fig.7[3].) library as well as to work into the ROS middleware, converting the images into ROS compliant messages.

[3]source: http://wiki.ros.org/vision_opencv

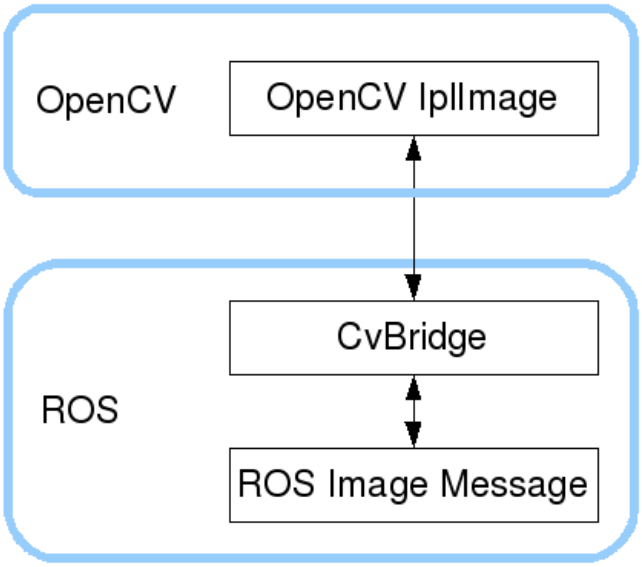

Figure 7: cv_bridge integration with the ROS middleware.

After feeding the node with the visual information, it runs a detector developed on top of the Tensorflow Object Detection API which leverages a pre-trained model to detect and recognize objects.
Eventually, it publishes the information on a topic, i.e. `/object_detection/detections` which offer the information about the label, the score and the bounding box.
At this point we have designed an adapter node which we believe it has been an helpful component pattern playing the role of making the final output information plainer and simpler.
In particular it reads the information from the `/object_detection/detections` topic and, according to a threshold set by the user's end on the score, outputs the final `/adapted_message` as a topic which is formalized as a string variable with the label of the object recognised.

2.2.3 ROS Object Detection and Recognition

For the development of our module we have decided to exploit Tensorflow, which is an end-to-end open-source platform for machine learning.
In particular, we have used the Tensorflow Object Detection API (see fig.8[4]).

[4]source:https: //github.com/tensorflow/models/tree/master/research/object_detection.

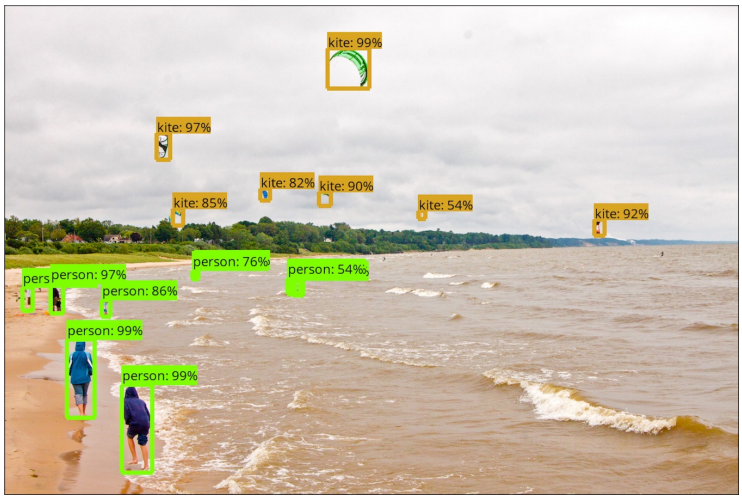

Figure 8: Tensorflow object recognition by means of the Tensorflow Object Detection API.

This latter is an open-source framework built on top of Tensorflow. Developed at Google, it offers the user the possibility to easily build, train and deploy object detection models. For the purpose we have leveraged a pre trained model: the SSD [4] mobilenet, whihch is a lightweight model able to run also on mobile devices, i.e. smartphones.
The SSD model is a feed-forward neural network which had been trained on the COCO dataset with a 6 layers architecture and for 90 different classes.

3 Implementation

3.1 Prerequisites

3.2 ROS

This project is develop using ROS, in particular it works on:

- rosdistro: Kinetic
- rosversion: 1.12

3.3 OS

The robot on-board is provided with Yocto Krogoth 2.1.1, an open source collaboration project that provides templates, tools and methods for Linux-based systems and embedded system deployments [5], and ROS kinetic. To access with a PC to MiRo, is fundamental to have installed only Ubuntu 16.04 LTS with ROS kinetic. No other version of both are stable, an updrade with Ubuntu 18 and ROS melodic is possible but it creates a lot of troubles during the configuration phase.

3.4 How to run

3.4.1 MIRO App

- Phone connected to the same WiFi as Miro
- Scan for MIRO
- Run/Restart Button

3.4.2 Terminal (Miro side)

- `ifconfig`, find your IP
- `ssh root@130.251.13.XXX`, search XXX on the MIRO App
- Insert Miro password
- `nano /.profile`
- search ROS IP and change it with your personal IP
- `source /.profile`

3.4.3 Terminal (PC Side)

- `cd ../catkin_ws`
- `catkin_make`
- `source ../catkin_ws/devel/setup.bash`
- `roslaunch ../catkin_ws/src/cob_people_object_detection_tensorflow /launch/cob_people_object_detection_tensorflow.launch` [6]
- `rostopic echo /adapted_message`

4 Results

We tested our project both in simulation and with the real robot. However, this results are only referred to the simulation scenario. The only change needed to bring the simulation code to the real environment is to change `camera_namespace` in `../launch/cob_people_object_detection_tensorflow_params.yaml` with the real robot camera topic. To summarize:

Gazebo simulation: `"/miro/sim01/platform/camr"`

Real robot: `"/miro/rob01/platform/camr"`

The Arena was created with a few objects like sport balls, standing person, a chair and a coke can, distribuited as in fig. 9.

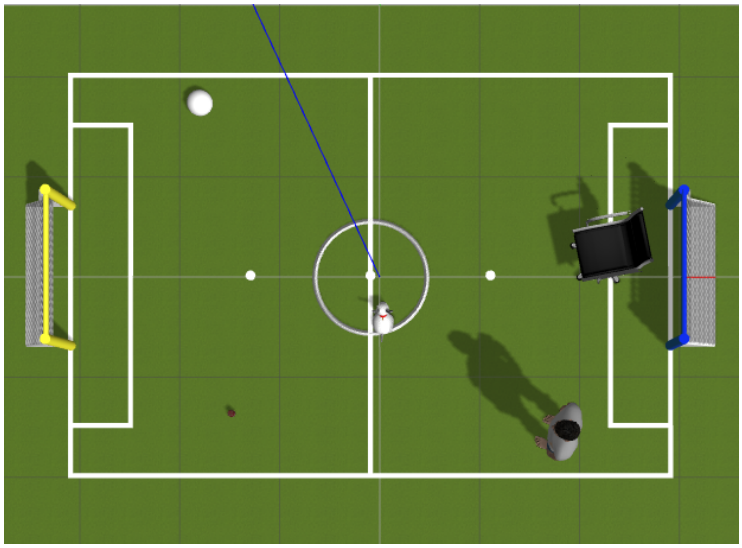

Figure 9: Gazebo Arena, objects distribution

In particular, the `/ir_adapter` node reads the information from the `/object_detection/detections` topic and, according to a threshold set by the user's end on the score, outputs the final `/adapted_message` as a topic which is formalized as a string variable with the label of the object recognised.
We have set the threshold on the score (bounded between 0 and 1) of the detected object to be 0.8, i.e. 80%. Whenever an object is detected with an accuracy bigger than 80%, the label is publish on the `/adapted message` topic. Miro Camera low resolution, lighting condition and movement affect the results' accuracy and contribute to errors in the detection process.
It's highly recommended with the MiRo in-eye camera to perform the code in a static situation and into a well-lit environment. Early test with MiRo in motion or in darker scenarios has proved us poorer results.

Here the video with the simulation, the GUI and in the bottom right corner is displayed the `adapted_message` topic.

The robot is able to recognize chair, person and sport ball.

5 Recommendations

It is mandatory to have the correct OS version and ROS version aforementioned in section 3. Guidelines for installing MiRo and ROS configuration are readable here
https://consequential.bitbucket.io/Developer_Preparation.html

6 Documentation

This node is based on a previous project of Cagatay Odabasi found on GitHub, called "ros_people_object_detection_tensorflow" accessible here https://github.com/cagbal/ros_people_object_detection_tensorflow.

6.1 Team

- Project Git Repo: https://github.com/EmaroLab/catkin_workspace_SOFAR_semantic_slam/tree/Module3
- Mario Ciranni: mario.ciranni@gmail.com, https://github.com/MarioCiranni
- Filippo Gandolfi: filippo.gandolfi59@gmail.com, https://github.com/filippogandolfi

References

[1] Consequential Robotics. *MIRO Beta Developer Kit*. http://consequentialrobotics.com/miro-beta. 2020.

[2] Raul Mur-Artal and Juan D Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[3] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[4] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[5] Richard Purdie et Al. *Yocto Project*. https://www.yoctoproject.org/. 2020.

[6] Cagatay Odabasi. *ros_people_object_detection_tensorflow*. https://github.com/cagbal/ros_people_object_detection_tensorflow. 2017.