Implementation of autonomous SLAM on a robotic rover using ROS

Guillaume Peter[1], **Tadej Petrič**[2] **and Andrej Gams**[2]

[1]*IUT of Cachan, Université Paris-Sud, 9 Avenue de la Division Leclerc, 94234 Cachan, France*
[2]*Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*
E-mail: guillaume.peter@u-psud.fr, andrej.gams@ijs.si

Abstract

In this paper we present an implementation of autonomous rover-robot navigation. This robot makes mapping using a SLAM system implemented with ROS. The robot can make a map of the environment and navigate in this environment without the intervention of a human. Initial implementation results are presented.

1 Introduction

Autonomous and mobile robots are the future of robotics in our world. To operate in the world, robots need to navigate in unknown environments. In order to locate themselves, robots typically use SLAM (Simultaneous Localization And Mapping) algorithms. Robots equipped with such technology are able to explore and understand a complex environment.

Programing of these robots is very complicated, as many things have to be taken into consideration. Building on previously developed and published tools makes it easier. This is the reason why we are using ROS (Robot Operating System) [1, 2]. This powerful and professional tool permits to make a link between simulation, programming and 3D visualization.

This paper presents the implementation of autonomous rover navigatoin behavior using a SLAM system within ROS. In Section 2 the meta-operating system called ROS is presented with its advantages and weaknesses. In Section 3 we discuss autonomous mapping with different approaches. Finally, Section 4 presents initial implementation results in the Piborg rover.

2 ROS

ROS is an open source software used to easily develop robotic applications. It was created in 2007 and since then the ROS community is getting bigger every year. It is being used by independent programmers and companies because of its compatibility. When you program with ROS you have the choice between C++ and Python and for the OS it's compatible with Linux, Mac OS and Windows.

When a project combines numerous micro controllers with a lot of different actions, their communication and integration quickly become overly complex. ROS simplifies all the communications between micro controllers and computers. Like this the calculation part is made by the computer and the micro controller makes only basic calculations. This environment is a multi-task environment with a very precise architecture. First, you need to run a master program which will setup the environment. Then we can run tasks in the environment. A task is called a node and it is used to communicate with another node. It can be used for different actions like read a sensor, control motors or make calculations. All the nodes can subscribe and publish topics. A topic is a named bus over which nodes exchange information. For example, a node which reads a temperature sensor will publish a topic with the value of the temperature. The big advantage of ROS is all the nodes can access to the topics, so data communications are easier.

But ROS doesn't have only advantages. This software need specific version of the OS. You can't adapt the version of ROS to your OS, you have to adapt your OS to the version of ROS. It's a very big forward jump concerning robotic interfaces but it takes some times to understand it clearly.

3 Autonomous Mapping

3.1 Hardware

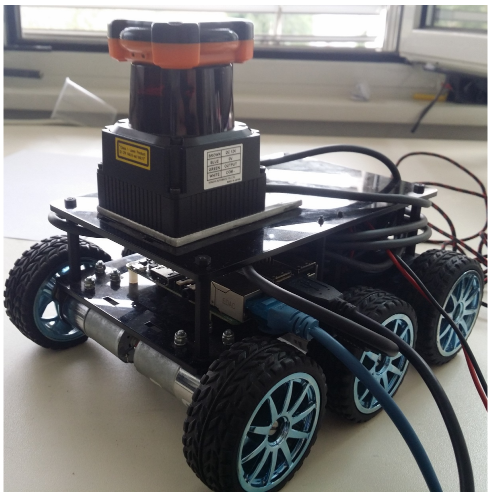

Figure 1: The small robot Piborg used in the project.

Figure 1 shows the robot we used for the project. To implement a SLAM algorithm we need a mobile robot. For the chassis of the robot we used a simple 6 wheels rover which turns by inverting the rotation of the wheels. Like this the robot can turn on the spot. Each wheels is equipped with a motor in order to have total control of the robot. To control the motors we use a Piborg card communicating in I2C with a Raspberry 2B, hence the name of the robot is Piborg [4]. This controller is a 2 channels controller. You can choose the way of rotation and the speed of rotation with it. In order to make the mapping we use a rotating range finder lidar sensor. It is situated at the top of the robot. This sensor is a distance sensor that measures distances on a plane around it. From the position of the sensor we can create a map with the collected data. We are using the Hokuyo UTM-30LX [3] which has a distance range of 30 meters and a resolution of 0.25 degree on 270 scanning degrees. It communicates by a USB cable with the Raspberry and the Raspberry is linked to the computer through a local network (for now ethernet cable, but we will make it wireless in the next implementation). We can summarize the connections with the schematic presented in Fig. 2. It shows the schematic of the connection between all the materials

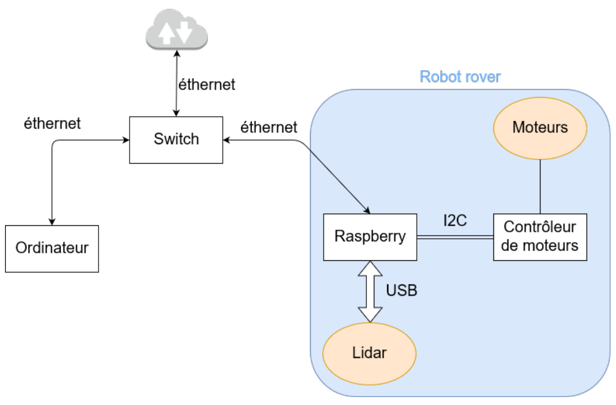

Figure 2: Schematic of the connections.

In Fig. 2, Ordinateur stands for computer, Contrôleur de moteurs is the Motor controller, and Moteurs are Motors. The cloud symbolizes Internet, because you need to be connected to Internet.

3.2 Software

Concerning ROS we are using ROS Kinetic which is known as a very stable version. On the computer we are using Ubuntu 16.04 and in the Raspberry we are using Ubuntu Mate. At the beginning of the project we first tried with Raspian for the Raspberry but quickly we concluded it was not appropriate for the application. In order to visualize the output of the sensor for mapping we use the vizualization software package Rviz.

3.3 SLAM

A robot equipped with a SLAM system is an autonomous robot. The objective is not limitied only to avoiding a wall. The robot must also understand its environment and how it can navigate in this environment. In the next section we give a brief overview about 2D navigation. Here we do not take in the consideration the z axis, since the robot navigates on a flat surface.

The first step is to create a map with the robot. The robot needs to know everything about its environment. It will explore and scan it with the lidar. Result of a scan is shown in Fig. 3. The software regroups all the collected data with the scans and builds a map in 2D.

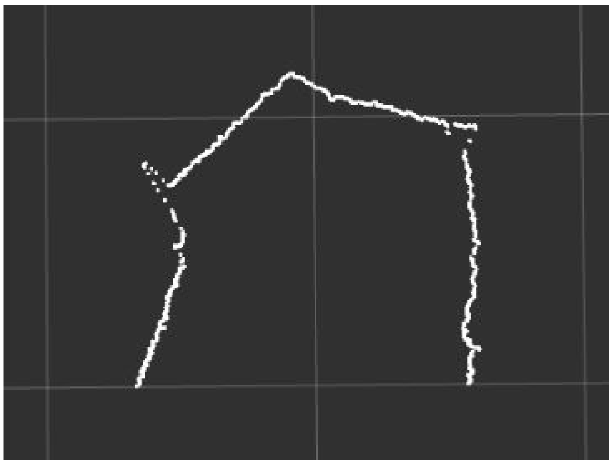

Figure 3: Result of a scan in Rviz

The robot needs numerous scans because like this it can confirm clearly the position of a wall or of an object. The robot needs to create a closed space, on the map there is no open door. To improve the quality of the mapping it's recommended to put encoders on the wheels.

Once it's made, the robot knows its environment but it needs to know its own position as well. It will navigate in a closed space with walls and objects, so the size of the robot is very important. A free space is not all the time a space where the robot can go. When the robot will move in the environment the lidar will scan all around it. Then the software will compare the obtained data with the map and find the position of the robot. This technique works better in an irregular space. In a square room with nothing on the ground, everything is too similar. Each scan has to indicate a unique space.

The SLAM method with ROS can be used easily because it can exploit the main feature of ROS - re-usability of robotic software. We just need to change some parameters. The tool named ”hector slam” [6, 5] permits to create maps and we can visualize it with Rviz. Odometry is taken in consideration by the software so it's very complete. But without odometry, if you don't have a referential point in a square room the localization doesn't work.

4 Results

In this part we discuss more about the project. What we have made, how we have made and how we want to exploit the project?

As you can see on the schematic in Fig. 4, our implementation relies on 3 nodes and 2 topics. We decided to separate the tasks to be more efficient.

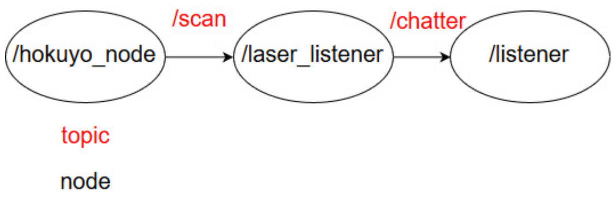

Figure 4: Schematic of architecture of the program.

The hokuyo sensor node is a task taken from the libraries of ROS. It permits to exploit correctly the lidar connected by the USB cable. This node publishes some topics, like the topic named scan. So the first node creates a link between the sensor and the Raspberry. The node named laser listener is a subscriber of the topic scan. It receives different data concerning the scans made by the sensor. Every 0.25 degrees made of the rotating sensor, the distance to objects around the robot is measured. Every revolution of the lidar the topic scan contains an array, each case corresponding to a distance taken at a precise angle. It is in the node laser listener we will detect objects in front of the robot. The robot needs to create a map and to not touch walls or objects. So if we see something in front of the robot it will turn in another direction.

But it is not the only task made by the laser listener node. The two nodes we have seen are implemented in the Raspberry. As said, the raspberry cannot make all the calculations to make the mapping. It would take too much time and we need a fixed screen where we can see the robot moving in the map. So we need to transmit all the data from the raspberry to the computer. For this we use an ethernet cable. Like this all the data are received on the computer and we can create and visualize the map in real time. The software Rviz is able to locate the topic scan as if it was running on the computer. It is one of the reason why ROS is very helpful. Once you are connected to the device you can visualize data as if everything is one system. The topic chatter permits to read the data of the sensor and the node listener permits to receive them.

Concerning the project, we are creating the map from the collected data as you can see in the Fig. 5

The final target of this is to put the robot in a room. Then with the SLAM system it makes a map of the room and we save it. But we want an autonomous navigation from the robot. Our goal is to put the robot in the room and on the software we select a point to go on the map. The robot should autonomously navigate to that position using the collected data from the sensor. Note that no odometry data will be present. The final goal is to build up the knowledge to implement similar autonomous behavior on a more advanced mobile robotic platform.

References

[1] ROS, http://www.ros.org/

[2] ROS,http://www.willowgarage.com/papers/ros-open-source-robot-operating-system

[3] Hokuyo, https://www.hokuyo-aut.jp/

[4] Piborg, https://www.piborg.org/

[5] SLAM, https://husarion.com/tutorials/ros-tutorials/6-slam-navigation/

[6] SLAM, https://www.mrpt.org/

[7] Ubuntu MATE, https://ubuntu-mate.org/raspberry-pi/

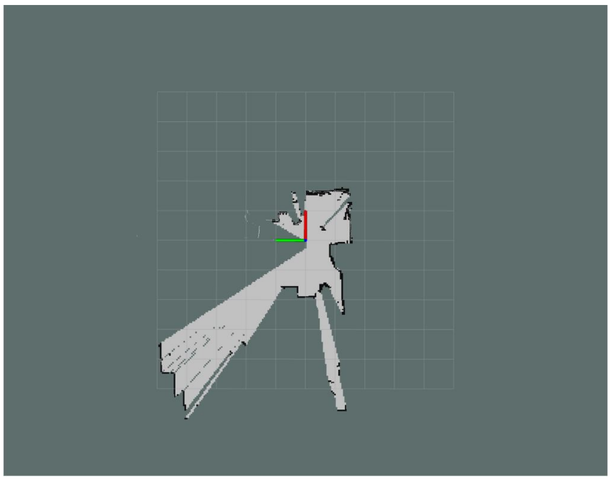

Figure 5: Visualization of a map on Rviz