TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
DEPARTMENT OF AUTOMATION

Project

Course: *Robotic Systems Control* – Professor: *Tassos Natsakis*
Due date: *January, 2022*

Background

About 14 years ago, a revolution on robotics started. It was eventually became what we know today as ROS (Robotic Operating System). ROS is not really an operating system, rather a collection of tools organised in such a way to allow researchers to develop and test new algorithms very easily. It has a modular design that allows to it expand in scope and its open nature makes it possible for anyone to contribute. Since this year, ROS2 is considered mature enough to be used in industrial applications and is expected to spark a new revolution on robotics and related fields.

That is why it is good for you if you learn something about it :)

Goal. The goal of this project is to teach you the basic concepts of ROS, and push you to experiment and develop your own applications. By implementation, ROS is suitable both for developing real-life applications but also constructing complex simulations to test algorithms and controllers. For this project, we will focus on simulations to simplify matters.

Your work will consist of the following aspects:

- Install and setup a ROS on your computer
- Learn about the basic concepts of ROS through tutorials
- Write your first simple applications
- Choose a robotic platform
- Write a real robotic application

The deliverable of this project will be a short report describing the robotic platform that you choose, the task that you developed, a performance analysis of your solution, and a video showcasing your application. This deliverable should be delivered by the end of the semester (week 14).

Until the delivery of the report, there will be three milestones that you need to report upon:

- Installation of ROS and completion of the basic ROS tutorials (Week 4).
- Presentation of your robotic platform (Week 8).
- Initial presentation of your task (Week 11).
- Final reporting (Week 14).

ROS Installation

ROS has been designed for linux systems and is heavily based on using the command line. There are installations for windows systems for the latest versions, however they are overly complicated and do not work as expected. Therefore, we will install ROS on a linux system. For those of you that do not have and do not want to install linux, we will use a virtual machine.

There are two major software solutions for virtual machines: VMware and VirtualBox. Both offer free versions of their software. If you want to quickly grab a ready-to-work virtual machine, for either VMware or VirtualBox, you can download your machine from `https://www.osboxes.org`. It is recommended that you install an Ubuntu 18.04 virtual machine.

Once you have your virtual machine setup, you should follow the Installation instructions for ROS Melodic (the latest long-term-support version of ROS). You might also want to learn a little bit about the linux command line, following these tutorials (I recommend going through Tutorial One, Two, Four, and Eight).

Tutorials and first application

Since ROS is a framework, it relies heavily on some concepts for its operation. Mastering these concepts is essential for working with ROS. Questions like 'What is a node?', 'What is a message?', 'What does it mean to subscribe to a topic?' should become clear after completing the basic tutorials of ROS. For the first milestone, you should complete the following tutorials of ROS:

- Installing and configuring your ROS Environment
- Navigating the ROS filesystem
- Creating a ROS package
- Building a ROS package
- Understanding ROS nodes
- Understanding ROS topics
- Using rqt_console and roslaunch
- Creating a ROS msg and srv
- Recording and playing back data

- Reading messages from a bag file
- Getting started with roswtf

The final tutorial, that will help you build your first basic ROS node, is the 'Writing a Simple Publisher and Subscriber'. You will notice that there are two versions of this tutorial, one for C++ and one for Python. ROS is designed to work with each of these programming languages. It is up to you to choose which language you will use for this project, based on your existing knowledge and/or career prospects.

You should complete all these tutorials latest by week 4 of the semester.

Choosing a robotic platform

One of the big advantages of ROS is the abstraction and modularity that it brought in the robotics world. The higher level tasks (task generation, path planning, user interfaces etc.) are totally decoupled from the lower-level ones (low-level joint control, communication with the robot etc. etc.). This allows us to develop applications that can be deployed on different robotic platforms with minimal changes in our code. This is possible as long as appropriate low-level controllers are existing for the respective robot.

Over the years, the community has written packages for the low-level communication and control of several industrial (and not only) robots. A non-exhaustive list of robots that packages exist for can be found on Robots ROS website. The website categorizes the robots based on their type (Aerial, Components, Ground, Manipulator, Marine), and the robots can be filtered based on different tags (research, 6 DOF, open hardware etc. etc.).

Your task is to choose a robotic platform of your liking, and making it work within your ROS workspace. It is highly recommended to choose a manipulator type of robot, so that you can apply the concepts that we learn at the course, but if you are very curious, you can choose a drone or a mobile robot as well. The task for this milestone is to install the necessary packages for the robot of your choice and to run a simulation where you can control the robot position manually using MoveIt!. More information will follow during the project hours.

Choosing a task

Having a robot in simulation is cool, but it is not useful by itself; we need to perform some kind of realistic task with it. This will be the most creative part of your project, in which you will implement a basic task to be performed with your robot. Here is a list of possible tasks, however if you have other ideas we can discuss them and decide what is possible.

- Pick and place an object from and to specific locations
- Perform trajectory planning around obstacles
- Perform trajectory planning and execution with waypoints
- Trajectory planning for two planning groups (end-effector, middle)
- Control the robot end-effector using the keyboard (6 DoFs)

- ...

Reporting

Please use the LaTeX fphw Assignment template for generating your report. Your report should include the following elements:

- A description of the robotic platform that you choose
- Links for all the resources that you used (libraries, robot description etc.)
- A description of the task that your robotic platform had to complete. This description should be as visual as possible
- A description of the performance of your implementation (how much time it takes to perform the task, what are the limitations, what are the specifications)
- A link to your video demonstration of your implementation

Grading and team-work

The workload of this project is meant to be for one person only, working for about one hour a week. Teams are not discouraged, however, if you decide to form a team, here are the rules:

- The grade of the project counts with 10% on your final grade
- Up to two people per team
- Each member of the team should complete the first two milestones (installation and tutorials)
- Each member should choose a separate task for their robotic platform. The tasks can be depending on one another, meaning that together they accomplish a more complex task
- Each member will be graded separately, based on their contribution to the task and their part of the final report
- On the last week, together with the report, you will have a short oral presentation of your solution
- If you do not present anything, you get a -1 penalty on your final grade.