RBE-550 Motion Planning

Daniel Montrallo Flickinger, PhD

Assignment 0

Software Installation and Getting Started

DUE: 2022-01-24 @ 12:00 UTC

Abstract

This assignment gets everything going with a running start for the beginning of the semester. First, make introductions to the class with an interesting motion planning application and a short summary about yourself and your interests. Then, get a development environment set up to use this semester, with Python, ROS, MATLAB, or any other tools. Use your environment to create an example two dimensional obstacle field. Finally, join the thriving demo scene and produce a quick demonstration of cutting edge turtle graphics.

1 Motion Planning Introduction

Find an interesting application of motion planning, and create a short report. Format may be written, slide presentation, video, or other. Cite references, and give clear examples. What do you find interesting about this method? What kind of problem does it solve? What familiar applications is it used in? If you don't have a favorite motion planning application, that's okay. Write up an epic rant and tell us all about how some aspect of motion planning is hopelessly broken and how only you can fix it.

Be brief, consider this an elevator pitch to the rest of the class. Post your application summary, along with a short introduction about yourself, to the course discussion board. Share your interests relating to engineering, motion planning, or robotics, as this will help everyone find group members for team assignments. And share something else unique about yourself. What's your favorite pizza topping? Who's your favorite muppet? Argue for Ford vs. Chevy vs. Tesla vs. VEB Sachsenring Automobilwerke Zwickau, etc…

2 Software Setup

No software installation is specifically required for assignments in this course, and students are welcome to undertake the arduous task of implementing everything manually using MIPS assembly or FORTRAN. For those of us with less time on our hands, there are many tools out there that give us a nice development environment for motion planning projects. Note that this course is not vocational training for ROS or MATLAB. These tools can be useful, but the topics of this course do not require them, or teach concepts in using them. That said, by all means post any questions and tips about software to the class discussion board. Also make use of external resources like Stack Overflow, as most problems encountered in this course should be common ones with readily available answers.

Work in this course should be largely cross platform, but for tools like ROS, setting up containers and virtualization helps. A containerization system such as Podman or Docker affords the ability to install dependencies for complex software without shuffling around system libraries. It also helps run Linux-specific software on other systems. For virtualization, VirtualBox is one option, among many others.

A Linux system, running Ubuntu 20.04 is the best recommended system for this course, whether on a laptop, in a virtual machine, or hosted in the cloud. Any other systems should work fine, thanks to the above mentioned options in containerization and virtualization.

In addition to a Unix-like system base, installation of Python for development is suggested. Consider using Python virtual environments (venv) for development work. There are a myriad of libraries available through pip to assist in build motion planning systems.

A MATLAB installation (available through WPI) is recommended, and could be useful, but is not specifically required.

Some assignments require video submissions, specifically screen recordings of simulations. An application such as OBS Studio is recommended for recording these videos. Use kdenlive for video editing.

As part of this assignment, set up your complete environment for this course. Minimize the number of tools requiring installation on demand later. Get set up with a revision control system, for you own use, and collaboration. Consider Gitlab at WPI, Github, or Bitbucket.

3 Turtle Graphics

To encourage the setup of a working Python environment, create a five point star using turtle graphics. Python has a built-in turtle graphics module, but students are welcome to boot up a LOGO floppy on their old IBM XT and plot out a star on the floor to be considered for submission, as seen in Figure 1.

4 ROS Installation

Create a working ROS installation for working in this course. Refer to the ROS Installation guide. The most recent long term support (LTS) release, Noetic, is recommended. If you are using a recent version of Ubuntu, a local installation is most straightforward. Follow the specific installation instructions. Otherwise, use a container-based installation. An additional option is to install Ubuntu 20.04 in a virtual machine and build a ROS installation there.

For more information, see the getting started guide and tutorials. Note that subsequent assignments will not necessarily require a ROS installation, but it might be useful. Also note that certain packages might require older versions of ROS. So consider spinning up a container running Ubuntu 16.04 with ROS Kinetic for certain tasks.

4.1 ROS Demonstration with TurtleSim

ROS includes a basic robot, Turtlesim, which is a small two-wheel mobile robot operating on a flat plane. Just like the turtle graphics robot in Python, and the LOGO turtle before it, basic planar geometry primitives are utilized by a virtual[1] robot to draw graphics. Various tutorials are well documented, and provide the opportunity to learn central concepts in ROS.

Navigate the turtle manually using keyboard control. Record the actions of TurtleSim and prepare a short video highlighting these tasks:

[1] or real!

Figure 1: WPI students utilizing LOGO Turtle robot to complete an RBE-550 assignment, c. 1978

- using ROS, run Turtlesim
- move the turtle using arrow keys to make some shapes (e.g., star, triangle, hexagon)
- echo its pose and command velocity
- EXTRA: change parameters, set a different channel background color, teleport the turtle to the netherworld

5 Create an Obstacle Field

Subsequent assignments will rely on planning within two dimensional *grid world* type environments. Using any tools or programming languages, create a random obstacle field in a 128×128 grid, using tetrominoes as depicted in Figure 2. Create a function, $f(\rho)$, to randomly distribute obstacles with varying coverage $\rho \in [0, 1]$. For example, a coverage rate of 10% would place approximately 256 obstacles in the field, occupying 1638 cells.

Demonstrate your obstacle field implementation by submitting full source code, and three figures depicting 10%, 50%, and 70% obstacle coverage.

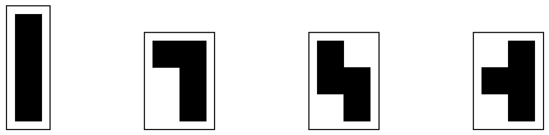

Figure 2: Example free tetromino obstacle shapes.

6 Grading and Submission

This assignment is due 2022-01-24 @ 12:00 UTC. *Late submissions are not accepted.* Upload completed assignment components (as individual files, not a single ZIP or TAR file) to the course site on Canvas.

Weight	Type	Component
20%	discussion board post	introduction, motion planning application
20%	PDF, or video	motion planning application
20%	source code, image	turtle graphics star
20%	video	recording of ROS TurtleSim
20%	source code, image	obstacle field demonstration

7 List of URLs

`http://wiki.ros.org/turtlesim`	p. 3
`https://arcgit.wpi.edu`	p. 2
`https://bitbucket.org`	p. 2
`https://canvas.wpi.edu`	p. 4
`https://docs.python.org/3.3/library/turtle.html`	p. 2
`https://docs.python.org/3/tutorial/venv.html`	p. 2
`https://github.com`	p. 2
`https://hub.wpi.edu/article/805/matlab-install-guide`	p. 2
`https://kdenlive.org/en/`	p. 2
`https://obsproject.com/`	p. 2
`https://pypi.org/project/pip/`	p. 2
`https://ros.org`	p. 2
`https://stackoverflow.com/`	p. 1
`https://wiki.ros.org/ROS/Installation`	p. 2
`https://wiki.ros.org/ROS/StartGuide`	p. 2
`https://wiki.ros.org/ROS/Tutorials`	p. 2
`https://wiki.ros.org/docker/Tutorials/Docker`	p. 2

Last update: January 12, 2022