ME132b — Lab 1

Vasu Raman, Scott C. Livingston

RELEASE: 3 APR 2015
DEADLINE: 10 APR 2015

Summary

Students will control a differential drive robot whose motion is constrained to have a minimum forward and maximum angular velocity. Students will make the robot visit waypoints in a space with static obstacles.

1 Objectives

- Recall creation of a simple ROS package.
- Recall running simulations in Stage.
- Implement a controller that steers the differential drive robot to visit a given list of waypoints.
- Experience the challenge of motion planning under constrained kinematics.

2 Logistics

This lab exercise makes use of the files in `me132a-final-20150306.tar`.

As in ME/CS 132a, you will have the option of remote access to an Ubuntu 14.04 (Linux) server that has all necessary software installed. Please contact Scott slivingston@cds.caltech.edu about getting an account. Alternatively, if you want to install it on your own, it should suffice to obtain the "desktop-full" distribution of ROS "Indigo"; consult instructions at `http://wiki.ros.org/indigo/Installation`.

An important part of this assignment is running your controller in the lab. You will need to schedule a lab session during the week of April 6 to do this. Announcements will be made during lecture or via the course website concerning available times.

If you have any requests or concerns with respect to lab accessibility, please contact us before your lab session. You may find additional helpful resources at `http://diversitycenter.caltech.edu` and `http://disability.caltech.edu`.

3 Background

Recall that a differential drive robot can simulate a unicycle, with the kinematics

$$\begin{aligned} \dot{x}(t) &= \nu(t)\cos(\theta(t)) \\ \dot{y}(t) &= \nu(t)\sin(\theta(t)) \\ \dot{\theta}(t) &= \omega(t), \end{aligned} \tag{1}$$

where $\nu(t)$ and $\omega(t)$ are control inputs, and $(x(t), y(t), \theta(t))$ is the robot's pose (2D position and orientation) at time t. Here $\nu(t)$ is commonly referred to as the forward velocity of the robot, and $\omega(t)$ the turning rate at time t. In this lab, we will control the Kobuki robots by directly commanding these velocities, instead of applying torques to the wheels separately.

4 Steps

4.1 Required

1. Let a and b be nonzero real numbers. Given constant control inputs $\nu(t) = a$ and $\omega(t) = b$ for $t \in [0, T]$, verify that the first two coordinates of the solution to the ODE in (1) from any initial state form a trajectory contained in a circle. Precisely, recall that for inputs ν, ω defined on the interval $[0, T]$, a solution of (1) is a function $(x(t), y(t), \theta(t))$ on $[0, T]$. Show that there exists $(c_1, c_2) \in \mathbb{R}^2$ and $r > 0$ such that for all $t \in [0, T]$, $(x(t) - c_1)^2 + (y(t) - c_2)^2 = r^2$.

2. Run `wander.py` or `wander.cpp` from `me132a_epoch` in simulation to remind yourself of what it does. Consult the `README` file in this directory for instructions.

3. Write a function to follow a sequence of waypoints while obeying each combination of the following constraints on the robot's forward and angular velocity:

$$\begin{aligned} \nu(t) &\geq \nu_{\min} \quad \text{such that } \nu_{\min} \in \{0.1, 0.2, 0.4, 0.6\} \\ |\omega(t)| &\leq \omega_{\max} \quad \text{such that } \omega_{\max} \in \{0.2, 0.4, 1, 1.5\}, \end{aligned}$$

   where the units of $\nu(t)$ and $\omega(t)$ are m/s and rad/s, respectively.

   Your function should take as input a list of waypoints provided as a plaintext file with one waypoint per row and space-separated x and y values for each waypoint. This list can be arbitrarily long and can contain duplicate copies of a particular waypoint. Waypoints must be visited in the order given. An example is included at the end of this assignment.

   After all waypoints are visited, the robot should come to a halt at the last waypoint.

You can assume the obstacles are known and static, but since the robot's position estimate is based on the robot's odometry and as such includes odometry errors, your code should provide some way for your robot to avoid collisions with the obstacles.

4. Demonstrate your controller in simulation prior to beginning your lab session.

4.2 Optional (Visualization)

In both simulation and the lab sessions, the following may be useful or fun, but are not crucial for completion of the assignment.

1. In both the lab session and simulation, watch the robot pose and range finder data using `rviz` (`http://wiki.ros.org/rviz`), a tool for real-time visualization.

2. Capture a snapshot of the known frames and transforms using `view_frames` from the tf package (`http://wiki.ros.org/tf`).

   ```
   rosrun tf view_frames
   ```

5 Deliverables

Please submit the following.

1. Your response to the first item in the list of requirements.

2. the *complete* ROS package in which you implement control and mapping. You *must* include a `README` or other concise instructions for reproducing results in simulation. E.g., you could provide a list of terminal commands and a few words describing intent, to aid in debugging in case the commands fail.

3. A written paragraph summarizing your conclusions from the lab. Were there particular settings of minimum forward and maximum angular velocity that made traversing the series of waypoints you chose more challenging? How did these constraints impact the reach error you were able to achieve?

There is no need to write a lengthy lab report. Providing the above in a tarball suffices. You should include your email address or telephone number so that the TA can contact you in case your submission fails (apparently) prematurely. This is not required but provides a back-up in case you make a trivial but epic mistake.

6 References

6.1 Kinematics for Wheeled Systems

- LaValle, Chapter 13.1.2.

6.2 ROS, Stage, etc.

- Concepts of ROS – `http://wiki.ros.org/ROS/Concepts`
- The package that wraps Stage in ROS – `http://wiki.ros.org/stage_ros`
- ROS node driver for the Hokuyo range finder – `http://wiki.ros.org/urg_node`
- roslaunch file syntax – `http://wiki.ros.org/roslaunch/XML`
- rosbag, tools for logging – `http://wiki.ros.org/rosbag`
- Stage documentation – `http://rtv.github.io/Stage/`

6.3 Hardware

1. Kobuki by Yujin Robot – `http://kobuki.yujinrobot.com`
2. Hokuyo URG-04LX-UG01 range finder – `http://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx_ug01.html`

6.4 Unix terminal

1. The Command Line Crash Course by Zed A. Shaw – `http://cli.learncodethehardway.org/book/`
2. An introduction to using SSH by Caltech IMSS – `http://www.imss.caltech.edu/help/ssh`

6.5 Example input format for waypoints

```
2 2
4 -1
2.2 3
1 1
```