Foundations of Robotics, Fall 2018

Coding Project 2: Mobile Robot Kinematics
(CS 4750: 95 points, CS 5750: 100 points)

Due Friday, Sept. 21 at 3:00 PM

Objective:

Demonstrate an understanding of mobile robot kinematics, specifically the steering mechanisms and the limitations of each one.

Collaboration Policy:

This assignment can be discussed as a group of arbitrary size. You may work with up to one partner on this assignment, but each student is responsible for writing and submitting a separate solution, and **no student should see another student's solution**. Include in a comment at the top of each file your name and the names of all classmates you discussed any part of the project with.

Using the Simulator Tool:

To run code for the assignment, use the command `./simulator-tool run p2 problemX`, where `problemX` is either `problem1`, `problem2a`, or `problem2b`. Note that you must complete the launch file for each problem in order for your code to be executed.

If you encounter difficulty using the simulator tool, please contact the course staff. Note that you can also run `./simulator-tool --help` to get a listing of commands and `./simulator-tool <command> --help` to get more information on the usage of a specific command.

Once you run a problem with the simulator tool, the V-REP simulator will open up. **To start the simulation, you must hit the play button.**

Debugging Tips:

If you add `output="screen"` to your launch file after you specify `type="something.py"`, the print statements from that Python file will print to your terminal. We recommend using `rostopic echo` to see what messages are being published; refer to the ROS documentation for more information.

If you would like to compare your published messages to the solution, you can use the provided solution topics. For each topic that your code will need to publish to, there is a corresponding solution topic which has the same name as the topic you need to publish to with `/solution` at the end. You can compare what the two topics are publishing by using `rostopic echo <topic name>`, or by using `rqt_plot <topic> <topic/solution>`.

If you try to retrieve of the provided constants from the ROS Parameter Server and find that it is not yet available, try retrieving it from within your callback function. This should ensure that the Parameter Server is running by the time you try to access a value.

We will have a FAQ and clarifications post for this assignment pinned on Piazza, so please check that post regularly for updates and information.

Conversion Formulas:

Information on converting steering mechanisms can be found in the Mobile Robot Steering Kinematics section of the course notes. For your convenience, we have included the Ackermann and differential drive formulas for you in this document.

Ackermann

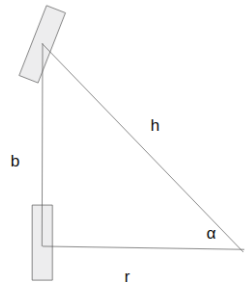

Figure 1: Ackermann steering

b is the base length of the Ackermann vehicle. κ is the curvature of the vehicle's path. r is the radius of curvature. α is the steering angle of the vehicle.

$$b = h\sin(\alpha)$$

$$r = h\cos(\alpha)$$

$$\omega = \frac{v}{r} = \frac{v}{b/\tan(\alpha)}$$

$$\kappa = \frac{1}{r}$$

Differential Drive

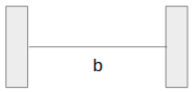

Figure 2: Differential drive

ω is the angular velocity. v_r is the velocity of the right wheel, v_l is the velocity of the left wheel. b is the distance between the left and right wheels. κ is the curvature of the vehicle's path. r is the radius of curvature. Note that v_r and v_l do not have an angular component

$$v = \frac{v_r + v_l}{2}$$

$$\omega = \frac{v_r - v_l}{b}$$

$$v\kappa = \omega$$

$$\kappa = \frac{1}{r}$$

$$\kappa = \frac{2(v_r - v_l)}{b(v_r + v_l)}$$

Assignment:

Implement code to solve the following problems, using the provided simulator framework. When you're finished, compress your complete ROS package with the command `./simulator-tool package p2`. Then upload the compressed package (named `p2_solutions.tar.gz`) to CMS. Note that you can upload to CMS as many times as you'd like up until the deadline; the most recent submission will override all the previous ones. Keep this in mind and don't wait until the last minute to submit! We recommend uploading at least a half hour before the deadline to ensure that your solutions are submitted on time.

Remember to run `./simulator-tool check p2` prior to submitting, which verifies that your submission is runnable and can be graded correctly. If this command returns errors or warnings, you will lose points on the assignment. Bear in mind that a successful return from the `check` command does not guarantee full points on this assignment.

Problem 1: Steering (50 Points)

This problem is designed to assess your understanding of different types of mobile robot kinematics. All of your code for this problem should go in the provided `src/problem1.py` file.

In this problem, you will subscribe to `p2/cmd_vel` for velocities of type `geometry_msgs/Twist`. Whenever the `p2/cmd_vel` topic publishes a message, your code will publish a message for each of the three robots listed below. More information on the content of the messages and where they should be published is detailed below.

You may find certain constant values are necessary to complete your computations. You can retrieve these constants using the ROS Parameter Server. For details on how to do this, please refer to the documentation. The parameter `base_length` is the distance between the front and rear wheels in the Ackermann car, and the parameter `diff_base_distance` is the base length for the Pioneer robot.

Most of `problem1.launch` is completed for you. You will need to add the line to run `problem1.py`. Refer back to the first coding project if you are unsure how to complete the launch file.

Omnidirectional

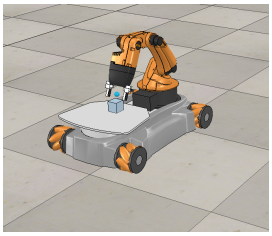

Figure 3: The KUKA youBot

In this scene, the KUKA youBot (shown in Figure 3) is your robot with an omnidirectional base. You will need to publish Twist messages to `vrep/youbot/base/cmd_vel`.

Ackermann

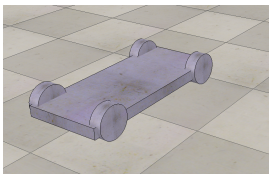

Figure 4: The Ackermann car

In this scene, the car model (shown in Figure 4) uses Ackermann steering. You will need to convert the provided velocity commands to the necessary format for the car and then publish the resulting Ackermann messages to `vrep/ackermann/cmd_vel`. For more information on the custom Ackermann message type, please refer to `p2/msg/Ackermann.msg`.

Differential Drive

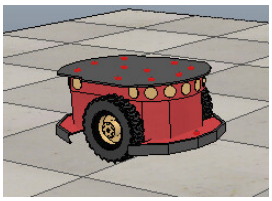

Figure 5: The Pioneer

In this scene, the Pioneer robot (shown in Figure 5) uses differential drive. You will need to convert the provided velocity commands to the necessary format for the Pioneer robot and then publish the resulting Differential messages to `vrep/differential/cmd_vel`. For more information on the custom Differential message type, please refer to `p2/msg/Differential.msg`.

To help you verify that the messages you are publishing are correct, we have created guide paths for you. If your robots are moving correctly, there will be a purple path right alongside the path of your own robot (shown in Figure 6).

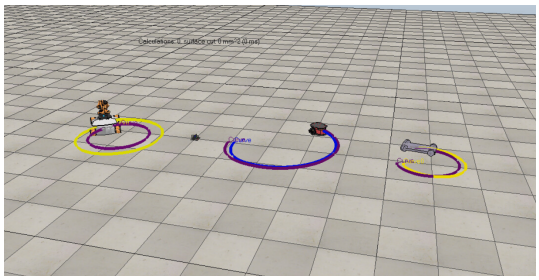

Figure 6: Guide paths for problem 1.

Steering Constraints

The physical constraints of a mobile robot will create different effects depending on the steering convention used. This problem is designed to explore the effects of some of these constraints on the steering of your robot.

Problem 2a: Ackermann (25 points)

In this problem, you have a scene with two Ackermann cars that have different base lengths. Ackermann messages that are used to drive the first car are published to the the topic `/p2/ackermann`. Whenever a message is published to this topic, you should determine an Ackermann message to be given to the second car such that the shape of the trajectories for both of the cars is about the same. The resulting message should be of Ackermann type and published to `vrep/ackermann/cmd_vel`.

There are some constants you can access using the ROS Parameter Server that you may find helpful in your computations. The parameter `base1_length` is the distance between the front and back wheels of the first car, and the parameter `base2_length` is the distance between the front and back wheels of the second car.

Your code for this problem should be written in `src/problem2a.py`. In addition, you will need to complete the launch file `launch/problem2a.launch` with the necessary information to run `src/problem2a.py`.

To help you verify that the messages you are publishing are correct, we have created a guide path for you. If your Ackermann car is moving correctly, there will be a purple path right alongside the path of the car (shown in Figure 7).

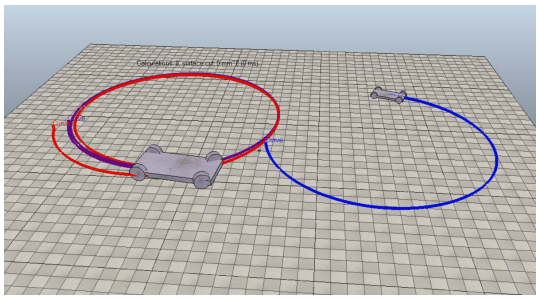

Figure 7: Guide path for problem 2a.

Problem 2b: Differential Drive (25 points)

In this problem, you have a scene with two Pioneer robots. The first robot has no limit on wheel velocities whereas the second does have a limit. Differential messages that are used to drive the first robot are published to the topic `/p2/differential`. Whenever a message is published to this topic, you should compute a comparable Differential message for the second Pioneer robot. The goal is for the second robot to have the same curvature as the first robot and velocities as similar as possible to the first robot. The result of your calculations should be published as a Differential message to the topic `/vrep/differential/cmd_vel`.

There are some constants you can access using the ROS Parameter Server that you may find helpful

in your computations. The parameter `diff_base_distance` is the base length for the Pioneer robot, and the parameter `diff_max_vel` is the maximum velocity for the wheels on the second Pioneer robot.

Your code for this problem should be written in `src/problem2b.py`. In addition, you will need to complete the launch file `launch/problem2b.launch` with the necessary information to run `src/problem2b.py`.

To help you verify that the messages you are publishing are correct, we have created a guide path for you. If your Pioneer robot is moving correctly, there will be a purple path right alongside the path of the robot (shown in Figure 8).

Figure 8: Guided path for problem 2b.