Lab 3: Inverse Kinematics

CSCI 3302: Introduction to Robotics

Report due Tuesday 2/25/20 @ 11:59pm

2 credit extra per day, up to 7 days/14 credits

The goals of this lab are to:

- Understand how to calculate the wheel speed given a desired velocity
- Experience basic feedback control
- See the need for higher-level reasoning / path-planning

You need:

- A functional Sparkiduino development environment
- A Sparki robot
- A working implementation of the Odometry Lab code
- The ArcBotics line following poster

Overview

A robot's forward kinematics allow you to compute the final position of a robot given individual joint positions. If we wanted to be able to perform the opposite process – figuring out the joint positions required to move a robot to a desired position/configuration, we will need to utilize *inverse kinematics*. A robot's inverse kinematics provide a joint configuration that will achieve a desired pose (in Sparki's case: x, y, θ). In this lab you will implement a feedback controller that navigates the robot from its current position to a provided goal position.

Instructions

Each group must develop their own software implementation and turn in a **single report** that contains:

- The code (1 .ino file)
- One individual lab report in PDF format. Please put the number of each question next to your answers, rather than turning in your answers as an essay

If your group does not finish the implementation by the end of the class, you may continue this lab on your own time as a complete group.

Part 1: Compute the Inverse Kinematics of Sparki

Use the forward kinematic relationship for a differential wheel platform from the book (x_r' and θ_r' as a function of left and right wheel angle change) to calculate its inverse, that is left and right wheel angle change given x_r' and θ_r'. **Note: you do not have to do the algebra yourself, but you should understand where the equations for $\dot{\phi}_l$ and $\dot{\phi}_r$ came from!**

Part 2: Position Controller

Helpful reminders: keep track of your units! I recommend using radians, meters, and seconds as your default rotation, distance, and timing units, scaling them as necessary for the functions you use: delay takes milliiseconds, sparki.moveForward takes centimeters, sparki.moveLeft/Right take degrees.

1. (Position Error) Calculate the Euclidean distance ρ between your current location and the goal position.

2. (Bearing Error) Calculate the angle α between the orientation of the robot and the direction of the goal position.
   For example, if the line between the robot's position and the goal position is oriented 90 degrees to the left of the robot, α is 90 degrees. If the goal is directly in front of the robot, α is 0 degrees (and so on). You should use the function atan2() for doing this.

3. (Heading Error) Calculate the angle β between the orientation of the robot and the goal orientation.

4. Using sparki.moveLeft, sparki.moveRight, and sparki.moveForward, create a controller that orients the robot toward the goal position (reduce bearing error to zero), drives to the goal position (reduce position error to zero), then orients to the proper heading (reduce heading error to zero). The robot's odometry can be updated all at once after these three actions are taken.

5. Verify that your controller works by driving to known locations around the map!

Part 3: Feedback Controller

This portion of the lab is more challenging than previous labs. Be sure to e-mail or slack your TA or your teacher with questions as they come up so you have plenty of time to solve the challenges you encounter!

1. Calculate a change in position x_r' that is proportional to ρ, e.g. $x_r' = 0.1 * \rho$
   What happens if you don't set/enforce a maximum value that x_r' can take in your IK code? Will your robot ever be able to rotate if $x_r' \gg \theta_r'$?

2. Calculate a change in rotation θ_r' that is proportional to α and θ_r, e.g. $\theta_r' = 0.1 * \alpha + 0.01 * \beta$ (make sure the direction of θ is consistent with your coordinate system)
   What should you do to α as the robot reduces its position error ρ near to 0? Should you weight it more heavily if ρ is large?

3. Create a proportional feedback controller that uses this signal to drive to a given goal. There are many valid ways to solve this problem. Describe your solution in your report.

You will likely have to update your odometry code to accommodate this new method of moving the robot! The function sparki.motorRotate(MOTOR, DIRECTION, SPEED) will be useful for this (http://arcbotics.com/lessons/moving-the-robot-home-lesson/).

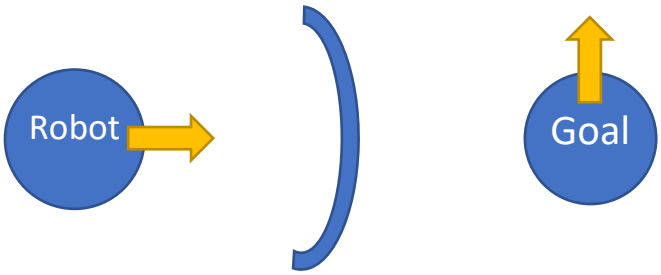
Keep in mind that the robot can only rotate its wheel a small amount during the 100ms loop delay, which is usually going to be less than ϕ_L' and ϕ_R'. You may wish to scale the SPEED arguments for each motor's motorRotate function call such that they are proportional to the ratio between ϕ_L' and ϕ_R'. (e.g.,: If $\phi_L' = 1.0$ and $\phi_R' = 2.0$, Right motor should run at 100% and left motor should run at 50%).

4. Verify that your controller works by driving to known locations around the map!

Part 4: Lab Report

Create a report that answers each of these questions:

1. Describe one condition where your controller from Part 2 might fail.
2. What is the role of the position error?
3. What is the role of the heading error?
4. Why include bearing error?
5. Describe how you implemented your feedback controller in Part 3.
6. Does your implementation for part 3 work? If not, describe the problems it has.
7. What are the equations for the final controller from your implementation? What are your equations for $\theta_r' \; and \; x_r'$ in your feedback controller?
8. What happens if you alter your gain constants (the 0.1 and 0.01 values in Part 3.1, 3.2)?
9. What happens if you increase these gain constants? What if they become too large?
10. What would happen if an obstacle was between the robot and its goal?
11. (Briefly) How would you implement simple obstacle avoidance using the ultrasonic sensor?
12. Describe what would happen if the obstacle-avoiding robot encountered a U-shaped object like this



13. What is the name of your team?
14. Roughly how much time did you spend programming this lab?

To be (optionally) submitted separately via e-mail to Prof. Roncone: aroncone@colorado.edu

(*Optional, confidential, not for credit*) A brief description of any problems (either technical or collaborative) encountered while performing this lab (e.g., issues with the clarity of instructions, clarity of documentation, lab colleague's behavior, etc.)