Project 1: Dead Reckoning and Tracking

3630: Introduction to Perception and Robotics
Mike Stilman

January 24, 2012

Administrative

- **Due Date:** 11:59pm Feb. 5
- **Deliverables:** Each group will email both TAs and Instructor ONE PDF describing your project as specified in the requirements. Someone in your group should know how to do this.
- **Demonstrations:** Jan. 31 - All the groups will demonstrate their projects. We will provide the overhead camera. You must get images from the camera and show that your algorithms work. (30% of project grade).

There exist two complementary methods for measuring the position of robots in the world: dead reckoning (odometry) and tracking. Odometry uses internal measurements of motor motion while tracking uses an external sensor. In this starter project, you will take an initial look at these methods, experiment with your robots and get familiar with our experimental setup.

1 Part I: Odometry

Your first task is to get your robot to move along a pre-specified trajectory: a square. Mark the starting point of the robot with some tape. Execute the following motions:

1. Move Forward $1m$
2. Turn Right 90 deg
3. Move Forward $1m$
4. etc. until you complete the square.

For both robots, make sure you do the turn. We do NOT require that the motions be precise or that your robot ends up at exactly the start. Each individual displacement should be within 30% of the target distance. It is your task to determine a mapping between the commands you send to the robot and the displacement that is achieved.

1.1 Questions

1. For each segment of your path, what is the difference between the target displacement and the one you achieved? Repeat the experiment at least 3 times and report averages.
2. When the trajectory completes, how far is the robot from its initial position and orientation?

3. What are the reasons for this error? Discuss the assumptions you made about how the robot functions and list any possible causes that it does not do exactly what you expect.

2 Part II: Tracking

The first part of this project has to do with internal sensing. Now, introduce the overhead camera as an external sensor. Place the camera $3m$ above the floor so that it has a clear view of the full workspace that will be used by the robot. (Note that its OK to use less or more distance at home. We will use a height of approximately 287cm for testing. During the live demonstration, we will be providing the camera for you to connect with USB and show us the following:

1. You may take a picture of the workspace before placing the robot.
2. After you place the robot in the workspace, run your trajectory and demonstrate the following:
   (a) Assume the robot starts somewhere close to the bottom-left corner of the image facing up in the image.
   (b) Locate the robot in the image and have your algorithm draw a square in your OpenCV window that represents the expected trajectory.
   (c) Track the robot by drawing a circle around it in the OpenCV window. The circle should stay around the robot as it moves.
   (d) Keep a list of robot positions as it moves and draw all these positions in your OpenCV window as small circles/dashes or lines to show its progress. This should result in a visible path that the robot has followed.

2.1 Questions

Assume that the camera is pointing straight down and the world frame is the point on the ground that shows up at the center of the camera image with X_w pointing right in the image and Y_w pointing up. Assume the robot starts somewhere in the lower-left corner of the image facing upwards (Y_w direction).

1. Use a homogeneous transform to represent the relationship between the world coordinate frame and the camera coordinate frame. Explain in words what this transform does. Why is it useful?
2. Use a homogeneous transform to represent the relationship between the camera frame and the image (pixel) coordinate frame. Explain in words what this transform represents. Why is it useful?
3. Give the homogeneous transform that maps world coordinates to image (pixel) coordinates.
4. Use the transforms above to draw a square in OpenCV that represents the target robot trajectory and include a screenshot in the report. Also, include a screenshot of the robot executing the trajectory.
5. What is the Euclidian distance (in pixels) between the robot's location at the end of the first segment of the trajectory and the predicted end of the first segment? Similarly, what is the distance between the robot's position of the whole trajectory and the expected location?
6. Identify the relationship between the errors in pixels and the errors you found in Part I.

3 Part III: Analysis

Having implemented all the steps in Parts I and II, let us consider a broader set of trajectories, analyze the challenges and think about solutions.

3.1 Questions

1. Design your own arbitrary trajectory for the robot to follow (be creative!). Repeat the steps from Parts I and II for this trajectory and show pictures that demonstrate the expected trajectory for the robot in OpenCV and the actual trajectory the robot followed.
2. Clearly, your robot never achieves the exact trajectory you expect. Lets think about what can be done to rectify this issue. Answer the following questions:
   (a) Suppose you can use the overhead camera while you control the robot. Can you get your robot to achieve each target point on the trajectory? If so, how? Provide less than 10 lines of pseudo-code that might be used to implement your solution.
   (b) Now, suppose we don't give you the overhead camera while you control the robot. Is it still possible to improve the accuracy with which your robot reaches the target points? If it is possible, describe a method that could be used. If not, explain why not.