Multi-Robots Systems : Laboratory Assignment 1

Nathalie Majcherczyk
Joseph Schornak
Nishan Srishankar
RBE 510 F16 - E. Eberbach
Worcester Polytechnic Institute

Fall 2016

1 Introduction

For this lab assignment, control and cooperation of multi-agent systems were introduced. The setup consisted of two Arcbotics Sparki robots (with disabled sensors), and an object that could only be maneuvered by multiple robots. A PID controller was used that obtained the position of the robots and entities via ArUco codes and a wide-angle camera.

The end goal of this lab assignment was to be able to detect the initial positions of the robot, the initial position of the box (which had two markers on it). After this was done, a series of translations and rotations would be implemented that would ensure the box was 2ft (x-direction), and 1ft (y-direction) away from the start position, along with a 90 degree change in orientation.

2 Perspective Correction

OpenCV's AR-code detection returns the coordinates of robots and other objects on the field in pixels. These coordinates do not uniformly translate to metric measurements because the camera is located off-center relative to the field, meaning that a vector in pixels could represent a dramatically different field-relative vector depending on its location in the camera image. This makes it difficult to accurately plan trajectories and conduct obstacle avoidance using only camera data.

We worked around this problem by creating a perspective transformation matrix through OpenCV to map pixel coordinates to metric coordinates. The function getPerspectiveTransform() takes in two arrays of four coordinates each. The first represents the detected locations of the corners of the field in the camera image, while the second represents the corresponding measured locations of the corner markers in centimeters. The algorithm calculates a transform matrix that maps the four input positions to the four output positions. This matrix can be used to perform the same transformation on any coordinates in the camera's field of view.

One drawback to this approach is that the transformation only works for objects in the plane of the table, so it is not possible to track objects in three dimensions. A consequence is that the difference in height between the AR markers on the surface of the table and the markers an the robots and boxes causes them to transform differently, making the perspective correction less accurate near the edges of the camera's field of view.

3 Path Computation

The path is computed and generalized for motion in every quadrant.The trajectory of the center of the box is obtained from the desired start and finish positions. Once this is computed, the path of the left and right robots are found via geometry. As can be seen in the following figures, the path computation is akin to that of differential steering on a

(two-)wheeled robot. The appendix shows some of the computations used for calculating the necessary paths. Once the paths for the robots are obtained, they are discretized by means of waypoints. Each waypoint for the left/right robots are then input to the given PID controller to be executed.

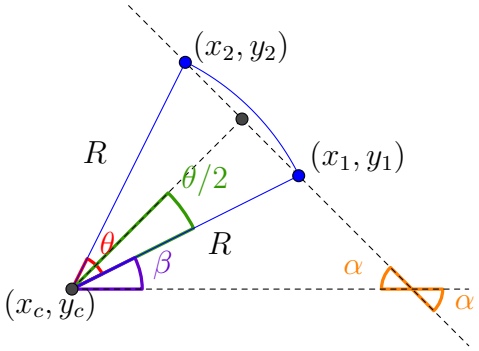
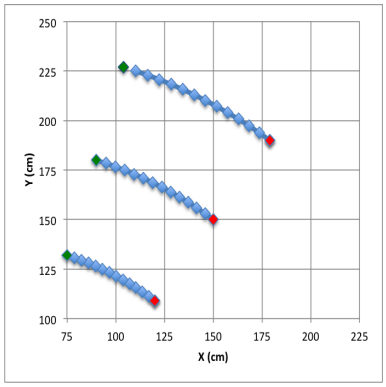
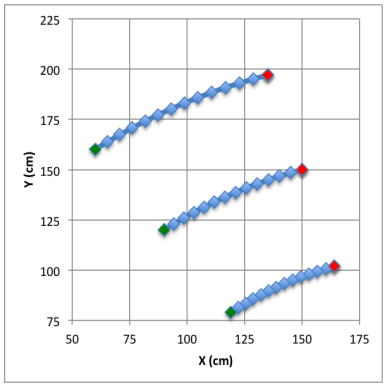


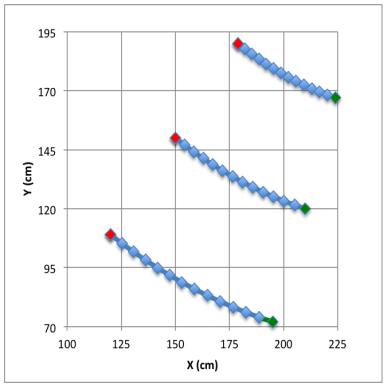
Figure 1: Basis of geometry used for path computation.



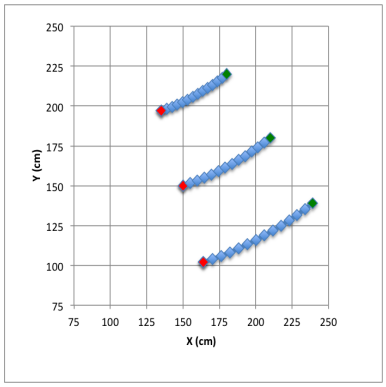
(a) Quadrant 1 circle arc



(b) Quadrant 2 circle arc



(c) Quadrant 3 circle arc



(d) Quadrant 4 circle arc

Figure 2: Counter-clockwise robot motion in each of the quadrants

4 Results

As can be seen in the following three figures, the actual trajectories of the center of the box, and the right/left robots are arcs with the same center of rotation. The actual positions of the robots closely match the specified waypoints given to them. The straight lines seen from when the experiment is started to when the robot starts to arc is due to the robot moving to the box as coded. The PID controller does the job of tracking its target quite well, with the exception of being incredibly slow. The video for this lab is uploaded on Youtube, titled "RBE510 Multirobot Systems Assignment 1"[1].

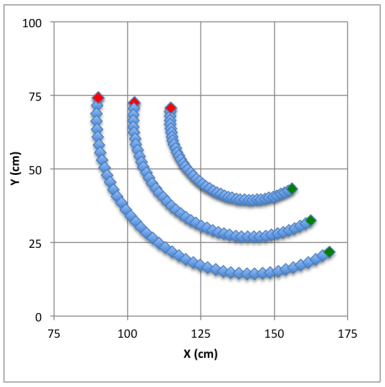


Figure 3: Computed paths (center of the box, right and left robots)

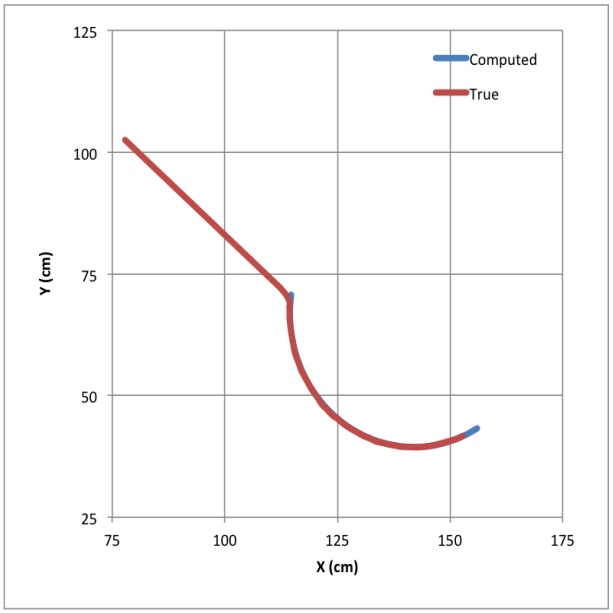


Figure 4: For right robot

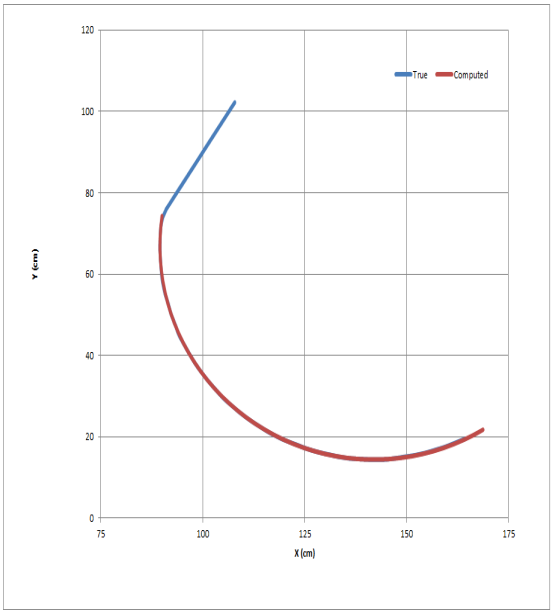


Figure 5: For left robot

5 Further Improvements

While the current algorithm for computing waypoints for each robot based on the desired trajectory for the center of the box is reasonably functional, it is not a completely generalized solution for the multi-robot path-planning problem. A more flexible solution could be based around quadratic Bezier curves. By setting control points P1 and P4 at the desired start and end positions, and setting control points P2 and P3 along lines intersecting P1 and P4 in the direction of the desired starting and ending orientation for each robot, the trajectory is guaranteed to be continuous through the entire length of the parametric curve while retaining desired orientations at each waypoint.

Another challenge not fully explored in this implementation is obstacle avoidance. To conduct certain types of maneuvers the robots would need to position themselves around each object to be moved while avoiding planning a path intersecting the object. An A*-based approach would require discretization of the navigable area and accommodation of a dynamic environment, so an alternate approach would be desirable.

Appendix

Quadrant 1

$$\alpha = -\arctan(\frac{\Delta y}{\Delta x})$$

$$\gamma = \beta$$

$$\beta = \frac{\pi}{2} - \alpha - \frac{\theta}{2}$$

$$x_c = x_1 - R\cos(\beta)$$
$$y_c = y_1 - R\sin(\beta)$$

$$x_i = x_c + R\cos(\beta + i(\Delta\theta))$$
$$y_i = y_c + R\sin(\beta + i(\Delta\theta))$$

Quadrant 2

$$\alpha = atan(\frac{y_2 - y_1}{x_2 - x_1})$$
$$\beta = -\frac{\theta}{2} + \alpha$$
$$\gamma = \beta + \frac{\pi}{2}$$
$$x_c = x_1 + R\sin(\beta)$$
$$y_c = y_1 - R\cos(\beta)$$

$$x_i = x_c + R\cos(\gamma + i(\Delta\theta))$$
$$y_i = y_c + R\sin(\gamma + i(\Delta\theta))$$

Quadrant 3

$$\alpha = -atan(\frac{y_2 - y_1}{x_2 - x_1})$$
$$\beta = \frac{\pi}{2} - \frac{\theta}{2} - \alpha$$
$$\gamma = \beta + \pi$$
$$x_c = x_1 + R\cos(\beta)$$
$$y_c = y_1 + R\sin(\beta)$$

$$x_i = x_c + R\cos(\gamma + i(\Delta\theta))$$
$$y_i = y_c + R\sin(\gamma + i(\Delta\theta))$$

Quadrant 4

$$\alpha = \arctan(\frac{\Delta y}{\Delta x})$$
$$\gamma = \beta$$
$$\beta = \frac{\pi}{2} - \alpha + \frac{\theta}{2}$$
$$x_c = x_1 - R\cos(\beta)$$
$$y_c = y_1 + R\sin(\beta)$$

$$x_i = x_c + R\cos(\gamma + i(\Delta\theta))$$
$$y_i = y_c + R\sin(\gamma + i(\Delta\theta))$$

References

[1] N. Majcherczyk, J. Schornak and N. Srishankar, "RBE510 Multi-robot Systems Assignment 1", YouTube, 2016. [Online]. Available: https://www.youtube.com/watch?v=NKM6iJk8OXo. [Accessed: 11- Nov- 2016].