CS 4970/5973
Intro Robotics
April 30, 2003

Amandeep Gill
Joshua Shuller
Celi Sun

Team 2 / Project 3

Robot Code Documentation

The code for project 3 was definitely not reactive, although some functions could be seen as being reactive. The code was more or less complete, without too many holes left open. And the robot performed consistently between runs. In the end, Team 2 scored 59 points.

Strategy for Achieving Goals:

The idea was to minimize the distance the robot had to travel in order to minimize error. So the code causes the robot to take each cube to the closest goal to that cube. And not only that, but the robot also grabs each cube from the closest goal to that cube. So the robot grabs all the cubes that are closest to the current goal and takes them back, until there are no more cubes that are closest to the current goal. After all the closest cubes to the current goal run out, the robot moves to the next goal to grab all of the closest cubes to that goal, and so on.

In short, the strategy implemented by the code caused the robot to do the following:

1. Go to the closest goal, ignoring all cubes.
2. Next, find the cubes whose closest goal is where the robot is. Grab the cubes that are on either the same x- or y-coordinate as the robot first. If there are no cubes on the same x- or y-axis as the robot, then just grab the closest.
3. If there are no more cubes whose closest goal is the current goal, move to a different goal and start out anew from that goal.

The robot gave priority to a cube if it was shared the same x- or y-coordinate as the robot. This way, the robot could align itself with the black tape surrounding the goal before grabbing it, since it was given that the goals would be square and at right angles with the rest of the arena.

This strategy was coded just a few hours before the demonstration. We spent most of our time working on precisely turning the robot with the CMUCam mounted on a servo, and on getting the robot to turn the direction we thought we told it to turn.

Tactics for Achieving Goals:

From the beginning we decided that precisely turning was the biggest problem of this project. We considered several options, but in the end decided that the CMUCam was the best bet. The idea here was to lock the CMUCam on a particular color blob, and turn using that color blob as the point of reference. After we got it to work, it turned out that it was more precise than using the encoders. We initially planned on keeping the CMUCam level with the ground so that we could use it to turn whenever there was a cube within its field of view. For tracking cubes close by, we had planned on using the `setWin()` function to only see cubes that were close by. However, the `setWin()` function seemed to have no effect whatsoever on the CMUScam, so we had to tilt the camera downwards so that it could only see within 1.5 feet of the robot. As a result, we could barely ever use the camera to turn, and we began depending more and more upon the encoders for direction detection, even though the encoders are not as accurate. And the primary purpose of the CMUCam became to detect if there was a cube in front of the robot or not.

The encoders were also used for going straight and detecting how far the robot had traveled. And the reflectivity sensors were used to detect the black tape surrounding the goals. Whenever the robot entered into a goal, it would first go inside the goal, and then turn to the nearest right angle. It would then align itself with the tape, count how far it had turned during its alignment, and correct its direction based on that. It would then backup, turn 90 degrees clockwise, and repeat the same thing again for the next side. And again, making it a total of 3 times that the robot aligned with the sides of the goal, so that by the time it was finished aligning, it was nearly certain that the robot was in the center of the cube, and that it knew its direction. This turned out to be the biggest advantage for achieving goals with our robot.

Conclusions:

In hindsight, it would have been better to not grab the cubes with the CMUCam at all, and only use the CMUCam for turning accurately. Another option would have been to use ET sensors and lever sensors to go along the walls and use the walls as the major point of reference.

It is not sound design to rely on either the CMUCam or the encoders for precise turning - it simply cannot be done without precise external reference. A more accurate way of tracking the outside world is required because the CMUCam API just doesn't cut it.