ECE 3090 Fall Semester, 2012

PROJECT 4 – ROBOT VISION-BASED LOCALIZATION

Assigned: Oct 19, 2012 Due: Nov 2, 2012 – 12:05pm

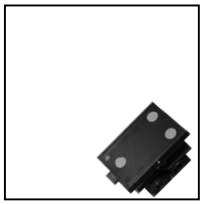
Thanks to your brilliant engineering designer, your robot explorer, sent to the most boring 2-dimensional planet in the known universe, was able to navigate from its current position to any other position on the planet. Unfortunately, you now realize that you have no idea how to determine your starting position. You are now a LOST robot explorer. Luckily, you have access to satellite imagery that can help you calculate your position and orientation.

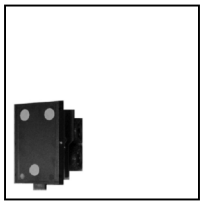
Input

Obviously, for your localization algorithm to function correctly, it needs overhead camera data. The input will be a series of image file names, each on a separate line, read from the file "input.dat". Each file represents a different position and orientation of the robot. The series of images will be in .pgm format. The final file name will be followed by a single line, "ENDOFINPUT".

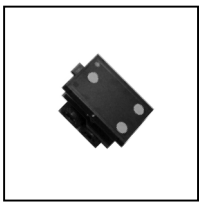
TestImage0.pgm
TestImage1.pgm
TestImage2.pgm
TestImage3.pgm
TestImage4.pgm
ENDOFINPUT

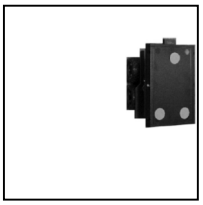
Here are the corresponding image files referenced in the example "input.dat" and provided on t-square.


TestImage0.pgm


TestImage1.pgm

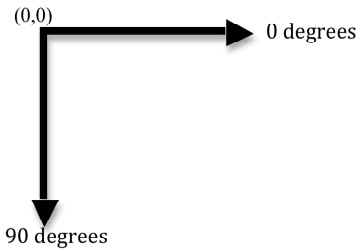

TestImage2.pgm


TestImage3.pgm


TestImage4.pgm

Output

For each image file, your program must print the (x,y) location associated with the center and the orientation of the robot, each on a separate line in the file "output.dat" (Note: All numbers must be positive). Robot location must be provided as pixel coordinates and orientation in degrees. Assume that your coordinate system is based on the representation below. Based on this construct, your robot would have an orientation of 180 degrees and a location of approximately (165, 44) in TestImage2.pgm.



Grading Policy: Although there are no requirements on design of the algorithm, you must use classes for this program. The code for each class definition must be stored in a separate file (i.e. Class1.cpp, Class1.h). The project will be graded based on the following criteria.

Optimal credit (20 points for each case): Program outputs a correct orientation within 10 degrees and pixel coordinates within a total error of 10 pixels. The program must run within 1.5 minutes.

Suboptimal credit (16 points for each case): Program outputs a correct orientation within 20 degrees and pixel coordinates within a total error of 30 pixels. The program must run within 1.5 minutes.

Partial credit (10 points for each case): Program outputs a correct orientation but not the correct pixel coordinates (or vice-versa). If you believe your program will take longer than 1.5 minutes to run through an image, you must indicate that via a message to the console.

***Additional Deductions**:*

- Program will not compile (60 pts)
- Program crashes or does not terminate (20 pts)
- Program does not contain at least one separate class file (20 pts)
- Program does not comply with requirements or good design constraints (e.g. non-working makefile, no zip file, inadequate comments, use of global variables, etc.) (2 to 10 pts)

Submission: Your project code is to be submitted via the ECE3090 T-Square site under Assignments-Project4 on or before the due date. All relevant files (including a makefile to compile your code) should be zipped up and named by firstName_lastName_project4.zip. We will test your code on the Jinx cluster so make sure your program correctly compiles and runs on that system. Information on the cluster is located at: http://support.cc.gatech.edu/facilities/instructional-labs/jinx-cluster.