Assignment 11: Shape Finder
CS 730/830, Fall 2016
Electronic submission due at 11:30pm on Mon, Nov 28,
Hardcopy submission due at start of class on Tue, Nov 29

Overview

You will write a program to find line segments in simulated laser rangefinder data given as an occupancy
grid. Graduate students will extend their program to also find circles. Your program will be given on
standard input a bitmap image (in ASCII format). It will emit the set of lines (and for grad students,
circles) that it found in the image. We recommend that you use the RANSAC algorithm. We provide an
image generator so that you can play around and tune your implementation’s parameters to achieve good
performance.

Input/Output

The bitmap will be provided on standard input in a format similar to the vacuum world (assignments 1
and 2) and motion planning (assignment 3) maps. All cells will be either empty (\_) or blocked (#). There
may be significant noise in the image data. Lines starting with / are comment lines and can be ignored.
Your program must emit all the circles it finds (zero of them, in the case of undergrads) and then all the
line segments it finds. Circles are specified by center \langle x, y \rangle, and radius, lines by the \langle x, y \rangle coordinates of the
endpoints. As in:

number of circles: 1
34.1 56.7 5.0
number of lines: 2
34.1 45.5 162.6 73.2
10.8 23.3 72.5 116.1

Coordinates are in map units (as in assignment 3), where \langle 0,0 \rangle is the bottom left corner.

Supplied Utilities

We supply:

image-generator generates sample data. Command line arguments include -he height -wi width -l
number-of-lines -c number-of-circles and -n number of noise cells. Try -h for details.

ransac-reference a sample solution.

ransac-visualizer a test harness to evaluate your program. It takes your program as a command-line
argument and the image file on standard input. It assumes that the image contains special comments
listing the shapes used to generate it and generates a picture comparing them to the shapes that your
program found.

Write-up

Electronically submit your solution using the instructions on the course web page, including your source
code as well as some pictures from your program running with the visualizer.
In class, submit a brief hardcopy write-up answering the following questions:

1. Describe any implementation choices you made that you felt were important. If you implement anything
beyond the assignment as written, please be sure to discuss it. Clearly explain any aspects of your
program that aren’t working. Mention anything else that we should know when evaluating your work.

2. What sorts of input will cause your program to fail? In other words, what does it have trouble with?

3. What suggestions do you have for improving this assignment in the future?
Design Suggestions

I’d suggest working first with images that contain only one simple shape, until your data fitting code works.

Appendix: Fitting Data

You’ll be dealing with numbers, so don’t forget to watch for things like division by 0. You should usually be able to deal with this by choosing new random points.

1. You will find yourself wanting to fit a line to two or more points. Many line fitting methods assume that, for particular known values of $x$, you have measured some $y$ values which might be corrupted by noise. Hence they find the line that minimizes the squared error in $y$ at each $x$ value. But in our situation, both the $x$ and $y$ values for a particular measurement could be subject to noise. We want to minimize the distance of each point from the fitted line, and this distance should be measured by the shortest distance to the line from the point (not just the distance in the $y$ direction).

The most robust way to do this is to use polar coordinates, but that gets complicated. In the name of simplicity, feel free to just use the equations for orthogonal regression on Wikipedia: the ‘Solution’ section under ‘Deming regression’, with $\delta = 1$. Note that $\beta_1$ is the slope of the line you want and $\beta_0$ is its $y$-intercept.

2. Given a line, you will want to compute the distance from a point $(x, y)$ to the nearest point lying on a line $y = ax + b$. This is $|ax - y + b|/\sqrt{a^2 + 1}$.

3. Now that you are in the midst of the assignment, you are probably realizing that what you really want is to fit a line segment to your data, not an infinite line. To extract a segment from your line:
   a) Project the inliers to lie along the line itself. For each point, the formula for the $x$ value is given in the ‘Deming regression’ article as $\hat{x}_i^*$ right after the line equations. Use the line to get the $y$ value for the point. Now that the points are on the line, we can represent segments by their start and end points.
   b) Sort the points (for example, by their $x$ values) and walk along them, splitting the line into segments whenever there is a large enough gap between points (say, for example, a distance of 3.5).
   c) Among all the resulting segments that have a sufficient number of inliers (say 7) and a sufficient density of inliers (say 0.7 inliers per unit length), call the one that explains the most inliers the best. If there are no such segments, then this line fit fails.

4. The RANSAC pseudocode from lecture recommends trying many fits and choosing the best. The value of ‘many’ might be 500.

5. Grad students will need to fit circles. To find the center $(x, y)$ and radius $r$ of a circle from three points $(x_i, y_i)$:

$$m_a = \frac{y_2 - y_1}{x_2 - x_1}$$
$$m_b = \frac{y_3 - y_2}{x_3 - x_2}$$
$$x = \frac{m_a m_b (y_1 - y_3) + m_b (x_1 + x_2) - m_a (x_2 + x_3)}{2(m_b - m_a)}$$
$$y = \frac{-1}{m_a} (x - x_1 + x_2) + \frac{y_1 + y_2}{2}$$
$$r = \sqrt{(x_1 - x)^2 + (y_1 - y)^2}$$

Don’t worry about refitting the circle to all the inliers. Finding the distance of a point from the circle is easy: find its distance from the center and compare it to the radius.

6. Feel free to tweak stuff to improve performance, but be sure to test on several different images to avoid overfitting.