Chapter 45

The Gauntlet

45.1 Overview

You have mastered the Bridge of Doom and navigated Flatland. Now you face the most challenging challenge you've ever been challenged with. In this final challenge, you will help your robot to dodge through obstacles on your way to the ultimate prize – knowledge.

Throughout this semester, you have applied many quantitative engineering analysis tools. For this challenge, you'll use a powerful new sensor (the laser scanner), explore some algorithms for optimization, and use the mathematics of potential functions and vector fields.

If you'd like you can work in a group (up to 3 people total per group) to complete the challenge and the write-up.

Learning Objectives

This challenge has varying levels of difficulty. The learning objectives vary based on the level you choose.

1. Fit models of lines and circles to laser scan data (level 2 and 3).
2. Implement outlier resistant optimization via the RANSAC algorithm (level 2 and 3).
3. Create suitable potential functions and vector fields to guid a robot to a goal while avoiding obstacles (all levels)
4. Transforming between multiple coordinate systems (all levels)

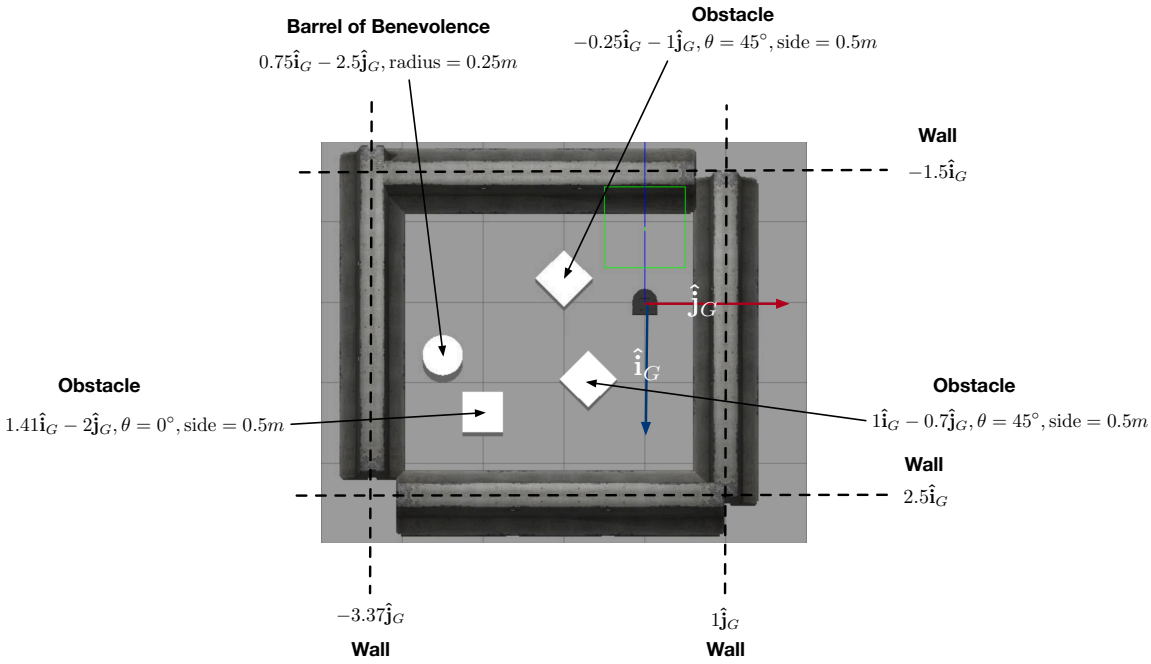


Figure 45.1: A top-down view of The Gauntlet. You should write a program to guide the Neato past the obstacles (boxes) to the goal (the Barrel of Benevolence).

45.2 The Challenge

Your goal is to write a program to pilot your (virtual) Neato through a series of obstacles (concrete barriers and boxes). You will detect these obstacles using your robot's onboard laser scanner, which can detect obstacles within a 5m range (see Figure 45.1).
This final QEA-I challenge has a hierarchy of missions, which get progressively more difficult. For all missions, the goal is to gently tap the Barrel of Benevolence (BoB). For all missions, the Neato starts at coordinate $0\ m\hat{\mathbf{i}} + 0\ m\hat{\mathbf{j}}$, facing along the $+\hat{\mathbf{i}}$ direction.
Here are the conditions for the missions:

Level 1 You are given the coordinates of the BoB and the obstacle and wall locations (see Figure 45.1). You will use these locations to create a potential field and use gradient descent to move your robot through the Gauntlet to the BoB. The path can be entirely pre-planned.

Level 2 You tackle the same challenge as level 1, but you must use the LIDAR to detect and avoid obstacles (i.e., you should not use the obstacle and wall locations in Figure 45.1, but instead determine them from the robot's sensors). You can plan your entire path based on your initial LIDAR scan or dynamically update your path as you go.

Level 3 You tackle the same challenge as level 2, but you are given the radius of the BoB (see Figure 45.1), but not the coordinates (i.e., you must use LIDAR scans to identify and locate the BoB). You can plan your entire path based on your initial LIDAR scan or dynamically update your path as you go. To

identify the BoB, you will need to differentiate the distinctive, circular shape of the goal from the linear shape of the obstacles. If you choose this mission, we recommend you look at the circle-fitting extension at the end of Day 9 for guidance.

To achieve whichever mission(s) you choose to accept, you will apply what you have learned about potential functions and vector fields. You may also choose to extend any of the above missions by applying or creating another goal-seeking/obstacle-avoiding algorithm, not taking the BoB's radius as an input, or trying to reach the target as soon as possible.

Loading the Gauntlet

Before starting the challenge, refresh your QEA robot software using the following command in Powershell (this will update the Gauntlet world).

```
docker pull qeacourse/robodocker:spring2020
```

In order to load the Gauntlet simulation, use the following Docker command. Once the simulator is up and running, you can program the robot as before in MATLAB.

```
docker stop neato; docker rm --force neato; docker run --rm --name=neato --sysctl
    net.ipv4.ip_local_port_range="32401 32767" -p 11311:11311 -p 8080:8080 -p
    32401-32767:32401-32767 -e NEATO_WORLD=gauntlet_final -it
    qeacourse/robodocker:spring2020
```

45.3 Deliverables

Don't think of these deliverables as only items to check off, but rather think of them as scaffolding to get maximum learning from the task at hand. By creating these intermediate deliverables, you will be able to more easily debug your code as well as your understanding of the algorithms you are utilizing.
Note that all of these deliverables can be done with your group (if you are working with others for the challenge). Make sure to clearly specify who is in your group for the challenge on each deliverable. Unless otherwise noted, the components are required for all levels of the challenge.

Mapping and Path Planning (due 5/7, but suggested completion date is 5/4) [6 pts]: We suggest that you make significant progress on the math side of this project before class on Monday. Create a document with the following plots:

1. A map of the pen, collected from a single LIDAR scan obtained from the robot's initial position, with walls and obstacles fitted as line segments using your RANSAC algorithm (levels 2 and 3). A fit for the BoB (level 3). [2 pts]

2. An equation and a contour (or 3-D) plot of the potential field you developed for the pen. The potential field can be based on the given features in Figure 45.1 (level 1) or based on a LIDAR scan from the robot's initial position (levels 2 and 3). [2 pts]

3. A quiver plot of the gradient of your potential field. [1 pt]

4. A path of gradient descent from the starting point to the BoB. [1 pt]

These plots should be clearly labeled and readable (decent size fonts!). **You will turn these in on Canvas through the Mapping and Path Planning assignment.**

Navigating the Gauntlet (due 5/7) [8 pts]: Prepare a video and a **brief** writeup of your work on this challenge.
Your final writeup should center around critical information and informative figures. It can be as short as you want, but it must contain the following components:

1. A short introduction stating the mission you chose and a brief summary of the strategy you used to solve the challenge. You might find it helpful to refer to the decomposition exercise from Day 9. [1 pt]
2. Some experimental data that shows, quantitatively, how well your system worked, plus a few sentences of explanation. Note: you could make use of the collectDataset_sim.m function for this.
    a) The time it took your robot to get to the BoB (for whichever mission you choose) and the distance it traveled. [1 pt]
    b) A plot with: a map of the relevant features of the Gauntlet based on a LIDAR scan (levels 2 and 3) or the features given in Figure 45.1 (level 1), the intended path of gradient descent, and the path your robot took calculated from the wheel encoder data. This should be a legible plot with axis labels, a legend, and a caption...the works. [3 pts]
3. A link to a video of your robot in action. [2 pts]

In addition to the writeup, you should also turn in your (commented and readable!) code. [1 pt]

Chapter 46

Day 9: The Gauntlet Challenge

46.1 Potential Fields

In the assignment you worked again on the concepts of scalar fields and vector fields, and you met the concept of "sources" and "sinks" of potential fields. Let's start by clarifying and synthesising some of this work. Recall that a **sink** at the origin is represented by the potential field

$$V(x,y) = \ln \sqrt{x^2+y^2}$$

which has the following gradient vector field

$$\nabla V = \frac{x}{x^2+y^2}\hat{\mathbf{i}} + \frac{y}{x^2+y^2}\hat{\mathbf{j}}$$

Exercise 46.1

1. What is the potential field and gradient vector field for a **source** at the origin?
2. What is the potential field and gradient vector field for a **sink** at the location (a,b).

Exercise 46.2

1. Use the annotation tools to sketch the contour levels for a **sink** at the origin.
2. Read the following MATLAB script and make sure you understand the purpose of every statement. Then implement it and compare the result with your sketch.

```
[x,y]=meshgrid(-3:0.05:3,-3:0.05:3);
v = log(sqrt(x.^2+y.^2));
contour(x,y,v,'k','ShowText','On')
axis equal
```

3. Write down the potential field for a **source** at $(1,2)$.
4. Use the annotation tools to sketch the contour levels with a **sink** at the origin and a **source** at (1,2).
5. Modify the MATLAB script above to confirm your prediction.

Exercise 46.3

1. Use the annotation tools to sketch the gradient vector field for a **sink** at the origin.
2. Read the following MATLAB script and make sure you understand the purpose of every statement. Then implement it and compare the result with your sketch.

```
[x,y]=meshgrid(-3:0.3:3,-3:0.3:3);
fx = x./(x.^2+y.^2);
fy = y./(x.^2+y.^2);
quiver(x,y,fx,fy)
axis equal
```

3. Write down the gradient vector field for a **source** at $(1, 2)$.
4. Use the annotation tools to sketch the gradient vector field with a **sink** at the origin and a **source** at $(1, 2)$.
5. Modify the MATLAB script above to confirm your prediction.

Exercise 46.4

1. Consider a source "line" sitting on the x-axis between x=-1 and x=+1. Use the annotation tool to sketch the contours of the potential field and the gradient vector field. (Hint: Start really far away, and then move in closer to the line.)
2. Read the following MATLAB script and make sure you understand the purpose of every statement. Then implement it and compare the result with the contours on your sketch.

```
[x,y]=meshgrid(-3:0.05:3,-3:0.05:3);
v = 0;
for a = -1:0.01:1
    v = v - log(sqrt((x-a).^2 + y.^2));
end
contour(x,y,v,'k','ShowText','On')
axis equal
```

3. How would you compute and visualize the gradient vector field for this line source? Go ahead and do so.

4. How would you change the source to a line from x=-2 to x=-1 at y=2? Go ahead and do so.

Exercise 46.5

1. Consider a source "circle" of radius 1 centered at the origin. Use the annotation tool to sketch the contours of the potential field and the gradient vector field (outside of the circle).
2. Read the following MATLAB script and make sure you understand the purpose of every statement. Then implement it and compare the result with the contours on your sketch.

```
[x,y]=meshgrid(-3:0.01:3,-3:0.01:3);
v = 0;
for theta = 0:0.1:2*pi
    a = cos(theta);
    b = sin(theta);
    v = v - log(sqrt((x-a).^2 + (y-b).^2));
end
contour(x,y,v,'k','ShowText','On')
axis equal
```

3. How would you compute and visualize the gradient vector field for this circle source? Go ahead and do so.
4. How would you edit the script to make the source be a circle of radius 2, centered at (-1,2)? Go ahead and do so.

Exercise 46.6

How would you define a general source (or sink) curve in the plane? Hint: use parametric equations!

46.2 Applying these ideas to the Gauntlet - Conceptual

Exercise 46.7

Now that you've thought about potential fields for sources and sinks (including lines and circles), we'd like you to conceptually think about how this might apply to helping a NEATO navigate the Gauntlet.

1. Review the Gauntlet—it can be found in chapter 45.
2. Thinking just about the walls, sketch the potential field and the gradient vector field if we treat the walls as line sources.
3. Thinking just about the obstacles, sketch the potential field and the gradient vector field if we treat the obstacles as being composed of line sources.
4. Thinking just about the barrel of benevolence, sketch the potential field and the gradient vector field if we treat the BoB as a circle sink.
5. Can you imagine how you would add all of the above sketches to build a sketch for the entire Gauntlet?
6. How would you use the computational tools from the earlier exercises to determine the actual potential field and gradient vector field for the Gauntlet?

46.3 Optional Extension: the Best-Fit Circle

If you plan to identify and locate the Bucket of Benevolence in The Gauntlet challenge, you will need to be able to find and fit a circle. These steps will guide you through the process of creating a circle-fitting algorithm of your own design.

Exercise 46.8

Assume that you have collected a set of N data points (x_i, y_i). There are multiple approaches to fitting a circle to this data, but we would like you to utilize the tools you have learned in QEA this semester, specifically linear regression and the RANSAC algorithm.

1. What are the different ways of describing a circle? Think about explicit, implicit, and parametric representations. Don't forget that there are various ways of describing a circle using each of these representations.
2. We would like you to set up the circle fitting problem using linear regression techniques from linear algebra i.e. an over-constrained system of linear equations where A*w = b. Choose one of the circle equations you identified and write out the A, w, and b matrices.

3. How would you solve this system?
4. Now imagine that you wanted to fit a circle using something more like a RANSAC approach. Describe the algorithm you would use, assuming that you knew the radius of the circle your were fitting (a la BoB). How many points would you need to select to fit each candidate circle?
5. What if you didn't know the radius? How would you have to modify your RANSAC approach?

Exercise 46.9

At this point, you have a couple of approaches in mind, and you've thought through the details of each. Let's test your ideas now by implementing in MATLAB.

1. Write pseudocode for finding the best-fit circle using one of your methods.
2. Generate some test points that lie on a circle of radius 2. Rather than generating points on the entire circle, let's generate points on an arc of 120 degrees to mimic the extent of the data we might get from the NEATO.
3. Test your methods on this data. How good is your best-fit circle? Do your results depend on which method you use, or how you initialize the method?
4. Now incorporate a little noise into your test data set by perturbing the radius of each of your data points (consider using the MATLAB function *randn* to do this). How does this affect your results? Make sure to try your code a couple of times to get a representative picture of how well it works on different noisy data samples.
5. Now implement the pseudocode you generated above into a working RANSAC-style circle fitting algorithm in Matlab. Test your code on the file playpensample.mat which has one instance of the BOB with r=0.1329m.

Solution 46.1

1. Include a negative in front to turn a sink into a source.
2. The potential field for a sink at (a,b) is

$$V(x,y) = \ln\sqrt{(x-a)^2+(y-b)^2}$$

which has the following gradient vector field

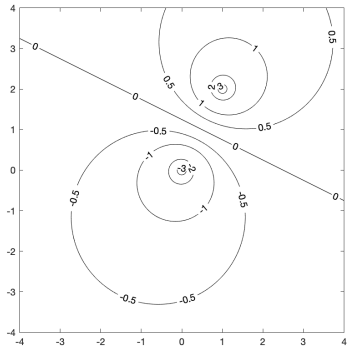
$$\nabla V = \frac{x-a}{(x-a)^2+(y-b)^2}\hat{\mathbf{i}} + \frac{y-b}{(x-a)^2+(y-b)^2}\hat{\mathbf{j}}$$

Solution 46.2

1. Circles surrounding the origin.
2.
```
%build a mesh
[x,y]=meshgrid(-3:0.05:3,-3:0.05:3);
%define the potential at all mesh points
v = log(sqrt(x.^2+y.^2));
%plot contours with levels marked
contour(x,y,v,'k','ShowText','On')
%so that circles look like circles
axis equal
```
3.

$$V(x,y) = -\ln\sqrt{(x-1)^2+(y-2)^2}$$

4. Circles when close to the sink or source. As you move away there are still circles but their center shifts. There is a line of constant potential that bisects the line that connects the sink and the source.



5.

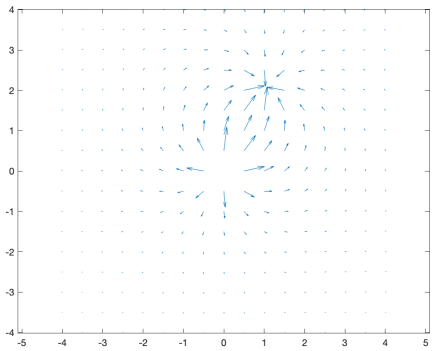
Solution 46.3

1. Arrows pointing radially outward.

2.
```
%create the mesh, less points so vectors visible
[x,y]=meshgrid(-3:0.3:3,-3:0.3:3);
%define the components of the gradient
fx = x./(x.^2+y.^2);
fy = y./(x.^2+y.^2);
%draw the arrows at the mesh points
quiver(x,y,fx,fy)
%so circles look like circles
axis equal
```

3. $$\nabla V = -\frac{x-1}{(x-1)^2+(y-2)^2}\hat{\mathbf{i}} - \frac{y-2}{(x-1)^2+(y-2)^2}$$

4. Arrows points away from the sink at the origin, and converging on the source at (1,2).

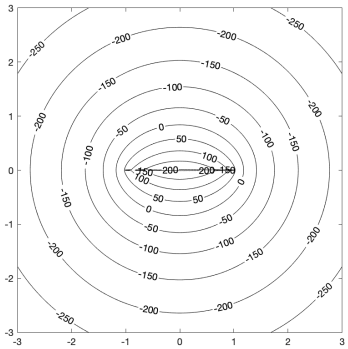
5. 

Solution 46.4

1. Far away the contours look like circles and the gradient field radiates inward to the source. Close to the line the contours have to be squeezed and will look more elliptical, with the gradient field radiating inward.

2.
```
%create a mesh
[x,y]=meshgrid(-3:0.05:3,-3:0.05:3);
%initialize the potential to zero
v = 0;
%loop through source points from -1 to +1
for a = -1:0.01:1
    v = v - log(sqrt((x-a).^2 + y.^2));%add the sources
end
```

```
%draw the contours
contour(x,y,v,'k','ShowText','On')
%so circles look like circles
axis equal
```



3.

4. We would sum up the gradient field from all of the sources on the line.

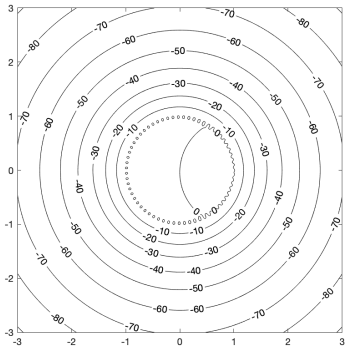
5. We would change the "for loop" as follows:

```
for a = -2:0.01:-1
    v = v - log(sqrt((x-a).^2 + (y-2).^2));
end
```

Solution 46.5

1. The contours will be circles, but they will decrease more slowly than the case of a single source at the origin. The gradient vectors will be radiating inwards.

2.
```
%create a mesh
[x,y]=meshgrid(-3:0.01:3,-3:0.01:3);
%initialize the potential to zero
v = 0;
%loop through angles from 0 to 2 pi
for theta = 0:0.1:2*pi
%sources have coordinates (a,b)
    a = cos(theta);
    b = sin(theta);
    v = v - log(sqrt((x-a).^2 + (y-b).^2));
end
%visualize the contours
contour(x,y,v,'k','ShowText','On')
%make circles look like circles
axis equal
```



3. We would sum up the gradient field from all the sources on the circle.

4. We would change the source coordinates as follows

```
a = -1+2.*cos(theta);
b = +2+2.*sin(theta);
```

Solution 46.8

1. What are the different ways of describing a circle? Think about explicit, implicit, and parametric representations. Don't forget that there are various ways of describing a circle using each of these representations. **Hint:** The general form of the circle equation will be helpful here.

   Implicit:

   $$(x-h)^2 + (y-k)^2 = r^2$$

   Parametric:

   $$x = h + r\cos t$$
   $$y = k + r\sin t$$

   General Form:

   $$x^2 + y^2 + Ax + By + C = 0$$

2. We would like you to set up the circle fitting problem using linear regression techniques from linear algebra i.e. an over-constrained system of linear equations where A*w = b. Choose one of the circle equations you identified and write out the A, w, and b matrices.

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} -x_1^2 - y_1^2 \\ -x_2^2 - y_2^2 \\ -x_n^2 - y_n^2 \end{bmatrix}$$

3. How would you solve this system?

   In Matlab we would use the backlash operator: $w = A\backslash b$; which solves the over-constrained system using linear regression.

4. Now imagine that you wanted to fit a circle using something more like a RANSAC approach. Describe the algorithm you would use, assuming that you knew the radius of the circle your were fitting (a la BoB). How many points would you need to select to fit each candidate circle?

   A circle fitting algorithm using a RANSAC approach could go like this:

   - Select three random points from the set. At minimum you need two points to fit the circle, but with only two points you have two solutions for the location of the circle center. Selecting three points removes this complication.
   - Using the three points, perform linear regression as outlined above to find the parameters A, B, and C of the candidate circle.
   - Find the radius and center point of the circle as: $x_c = -A/2$, $y_c = -B/2$, and $r = \sqrt{x_c^2 + y_c^2 - C}$
   - Find the distance d of each point x_n in the set from the candidate circle edge as: $d = \|\sqrt{(x_n - x_c)^2 + (y_n - y_c)^2} - r\|$
   - Determine the number of inliers as points that fall within a threshold distance from the circle edge.
   - If the candidate circle has the most inliers **and** has a radius close of the radius of the BOB, keep the candidate circle as the current best fit. If not, try again.
   - Repeat.

5. What if you didn't know the radius? How would you have to modify your RANSAC approach?

   Without knowing the radius, it would likely be helpful to reduce the threshold tolerance and use a larger set of sample points. This would help to identify structures that were actually circles, instead of fitting a circle with a very large radius to points that more accurately make up a line.

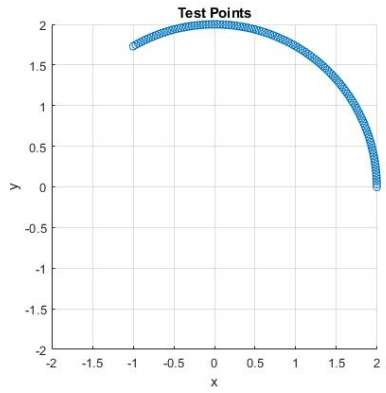
Solution 46.9

1. Write pseudocode for finding the best-fit circle using one of your methods.

   The proposed method above could be turned into pseudocode with additional detail and steps/function names.

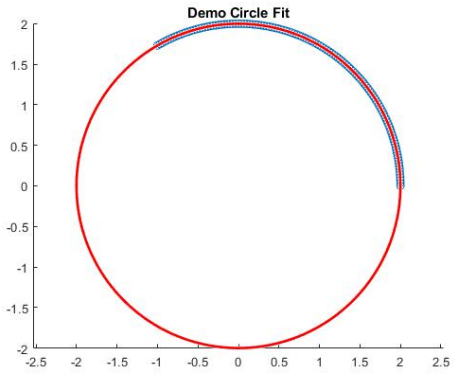
2. Generate some test points that lie on a circle of radius 2. Rather than generating points on the entire circle, let's generate points on an arc of 120 degrees to mimic the extent of the data we might get from the NEATO.

   The function here allows the generation of test points for a circle with a chosen noise value.



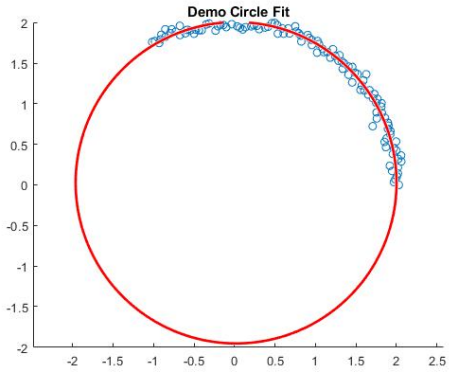
3. Test your methods on this data. How good is your best-fit circle? Do your results depend on which method you use, or how you initialize the method?

   The first section of the circle fitting starter code here demonstrates the application of the linear regression approach above to test data points.



4. Now incorporate a little noise into your test data set by perturbing the radius of each of your data points (consider using the MATLAB function *randn* to do this). How does this affect your results? Make sure to try your code a couple of times to get a representative picture of how well it works on different noisy data samples.

   Again we can try this using the circle fitting starter code here by adding noise to the test points.



5. Now implement the pseudocode you generated above into a working RANSAC-style circle fitting algorithm in Matlab. Test your code on the file playpensample.mat which has one instance of the BOB with r=0.1329m.

   A working circle fitting script with comments is here.