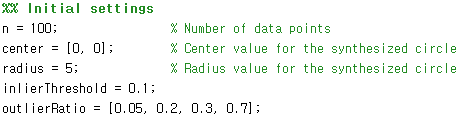
# EXERCISE 1: ROBUST ESTIMATION AND NORMS

GCT722 MATHEMATICAL METHODS FOR VISUAL COMPUTING  
20183151 Chaelin Kim

## PART 1: RANSAC FOR CIRCLE FITTING

### **Description of implementation**

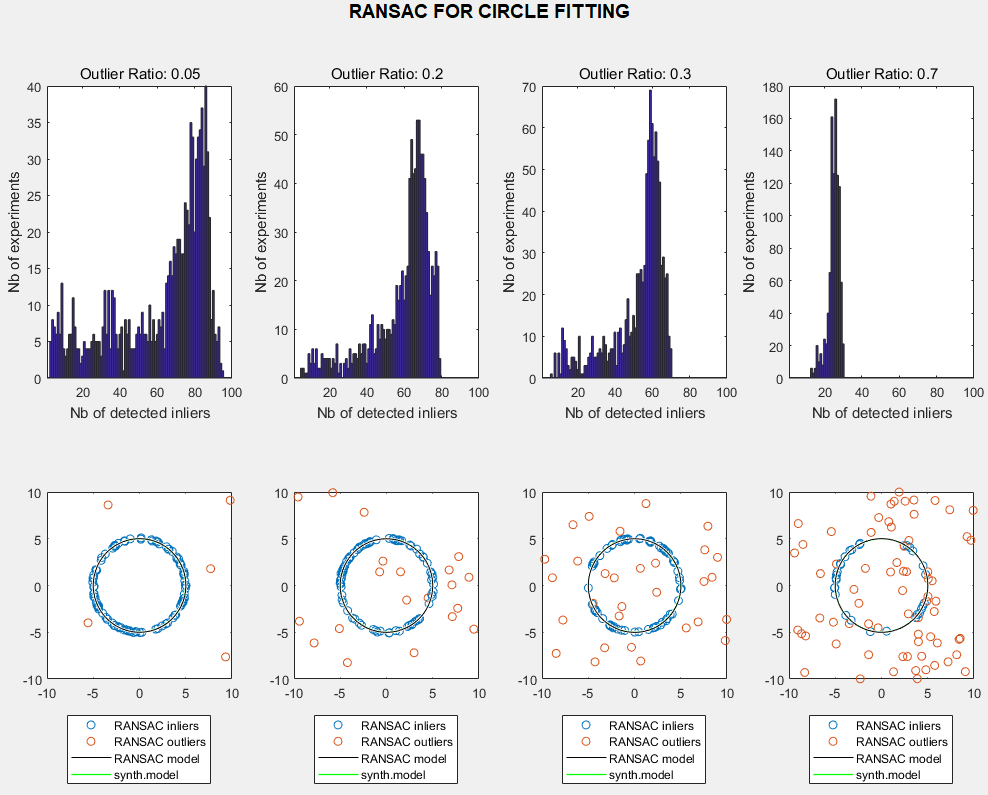
There are 1 script file & 4 function files for the exercise RANSAC.

* **Script file**
* **main\_RANSAC.m**  
  : This is the script for doing RANSAC for circle fitting. It calls functions for RANSAC (genCircleData, doRANSAC, doExhaustiveSearch, drawRANSACPlot). Details of these functions are explained below. We can set the initial settings in this script.  
  
* **Function files**
* **genCircleData.m**  
  : This function generates data points on a synthesized circle with inliers and outliers. The number of inliers and outliers are decided by the given outlier ratio. The inlier data is made with random noise (between -0.1 and 0.1). In the part of outlier generation, *while* loop is executed until the number of made outlier data meets the given number of outliers.  
    
  
  + Input Value
    - *n*: The number of data points
    - *center*: The center (x, y) value of the synthesized circle
    - *radius*: The radius value of the synthesized circle
    - *inlierThreshold*: The inlier distance threshold
    - *outlierRatio*: The ratio of outliers in data
  + Output
    - *data*: The data in the form of circle with inliers and outliers according to the outlier ratio.
* **doRANSAC.m**  
  : This function runs RANSAC for the given data. In this function, the number of RANSAC iteration is computed and used in *for* loop for RANSAC (How to calculate this number is explained in Questions category below (What about the number of RANSAC iterations with r = 5%, 20%, 30% and 70%?)). For doing RANSAC with the circle data, we can pick the random three points. The center of the circle passing through these three points is the intersection of the perpendicular bisectors of the lines connecting the two points. In calculating errors(distances), Inliers are calculated by the center and radius values of founded circle and saved in the matrix. Then the best model is updated by the number of inliers. This process is iterated 1000 times for finding best RANSAC model.  
  (Reference how to find a circle with 3 points: http://egloos.zum.com/heilow/v/418569)  
    
  
  + Input Value
    - *data*: The data made from genCircleData function
    - *M*: The number of Iteration for re-apply RANSAC
    - *inlierThreshold*: The inlier distance threshold.
    - *outlierRatio*: The ratio of outliers in data
  + Output
    - *bestModel*: The computed data [center\_x\_value, center\_y\_value, radius]
    - *detectedInliers*: The number of computed inliers of each iteration of re-apply RANSAC
* **doExhaustiveSearch.m**  
  : This function runs exhaustive search for the given data. The combination value is calculated by function *nchoosek*. The method how to find inliers is similar to *doRANSAC* function. In addition, it has the code for displaying the result of exhaustive searching.  
    
  
  + Input Value
    - *data*: The data made from *genCircleData* function
    - *k*: How many points are selected for searching
    - *inlierThreshold*: The inlier distance threshold
    - *outlierRatio*: The ratio of outliers in data
  + Output
    - No output but it displays the result on the command window.
* **drawRANSACPlot.m**  
  : This function draws histograms of re-applying RANSAC and plots of RANSAC results with different outlier ratios. The result is shown in category *Screenshots* below.  
    
  
  + Input Value
    - *center*: The center (x, y) value of the synthesized circle
    - *radius*: The radius value of the synthesized circle
    - *data*: All data made from *genCircleData* function (type: cell)
    - *ransacResult*: The results of RANSAC made from *doRANSAC* function (type: cell)
    - *histResult*: The results of the number of inliers made from *doRANSAC* function (type: cell)
    - *outlierRatio*: The ratio of outliers in data for draw outlier data
  + Output
    - No output. It shows result plots.

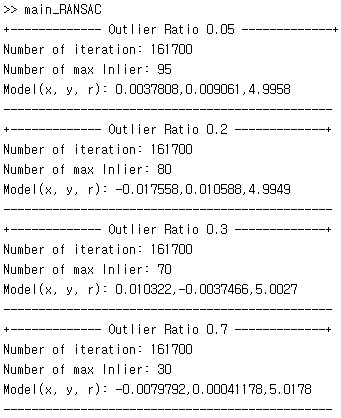
### **Instructions for running**

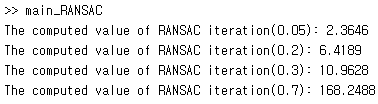
1. Open the file “main\_RANSAC.m” in Matlab.
2. Execute that file.
   1. You can see the result of exhaustive search in command window.
   2. The another window that shows plots of RANSAC results is opened.

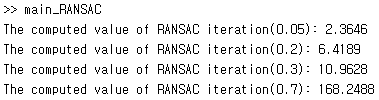
### **Screenshots**



### **Questions**

* **How many combinations (exhaustive search) exist for N = 100 points?**
* Ans: 161700  
    
   
* **What about the number of RANSAC iterations with r = 5%, 20%, 30% and 70%?**
  + - *p*: Probability that at least one sample has no outliers
    - : Outlier ratio
    - *s*: Sample size (minimum data)

🡪   
 

* **What about when N = 10,000 points?**
* Exhaustive search  
  combinations =   
  🡪 Very big number…
* RANSAC iterations 🡪 not changed!  
    
  🡪 The number of RANSAC iterations is not affected with the number of data because it is effected by precision probability, outlier ratio and sample size.

### **Discuss the results**

In this code, I made synthesized model with outlier ratios 5% 20%, 30% and 70%, did RANSAC and exhausted search with synthesized data, and draw the plots that show the results.

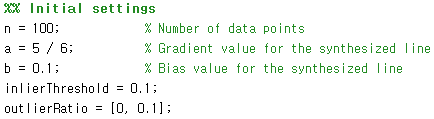
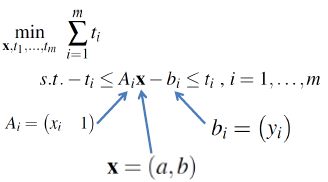
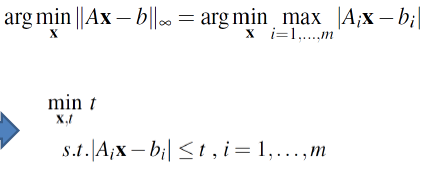
In the histogram, considering with outlier ratios, RANSAC finds the number of inliers similar to real number of inliers. In the plots, RANSAC with re-applying 1000 times finds quite fitted circle model in case of all outlier ratios.

Exhaustive searching finds the best fitted model because it considers all combinations of data points. In that point, if the number of data increases, the computational cost exponentially increases. On the contrary, RANSAC is not affected with the number of data. So, if other conditions are fixed and the number of data increases, RANSAC finds a fitted model faster than exhaustive search. But if iteration for re-applying is lower and outlier ratio is higher, RANSAC would be hard to find well-fitted model.

## PART 2: IRLS AND NORMS FOR LINE FITTING

### **Description of implementation**

There are 1 script file & 4 function files for the exercise RANSAC.

* **Script file**
* **main\_LineFitting.m**  
  : This is the script for doing line fitting using norm (employing IRLS and LP) and  norm (employing LP). It calls functions for line fitting (genLineData, doIRLS, doLP, drawLineFittingPlot). Details of these functions are explained below. We can set the initial settings in this script.  
  
* **Function files**
* **genLineData.m**  
  : This function generates data points on a synthesized line with inliers and outliers. The number of inliers and outliers are decided by the given outlier ratio. The inlier data is made with random noise (between -0.1 and 0.1). In the part of outlier generation, *while* loop is executed until the number of made outlier data meets the given number of outliers.  
    
  
  + Input Value
    - *n*: The number of data points
    - *a*: The gradient value of the synthesized line
    - *b*: The bias value of the synthesized line
    - *inlierThreshold*: The inlier distance threshold
    - *outlierRatio*: The ratio of outliers in data
  + Output
    - *data*: The data in the form of line with inliers and outliers according to the outlier ratio.
* **doIRLS.m**  
  : This function runs IRLS with for the given data. For IRLS, we change norm to weight \* norm for easy computation. norm is computed with setting the gradient to 0. The weight is computed with the value of gradient and bias calculated by norm. With this weight, the result is recomputed. This process is iterated until the absolute value of difference between previous value and current value is smaller than 0.0001.  
    
  
  + Input Value
    - *data*: The data made from *genLineData* function (type: matrix)
  + Output
    - *result\_IRLS*: The result data [a, b] computed by IRLS
* **doLP.m**  
  : This function runs Linear Programming(LP) with  and  for the given data. Linear programming solver finds the minimum of a problem. The purpose of is minimize the sum of the absolute value of all errors and the purpose of is minimize the maximum error among the absolute value of errors. In order to solve this problem with LP, I made the matrix A and b that consist of linear equations with given data. Using *linprog()* function, the result is easily computed.  
  `   
  🡪 for LP(left) and for LP(right)

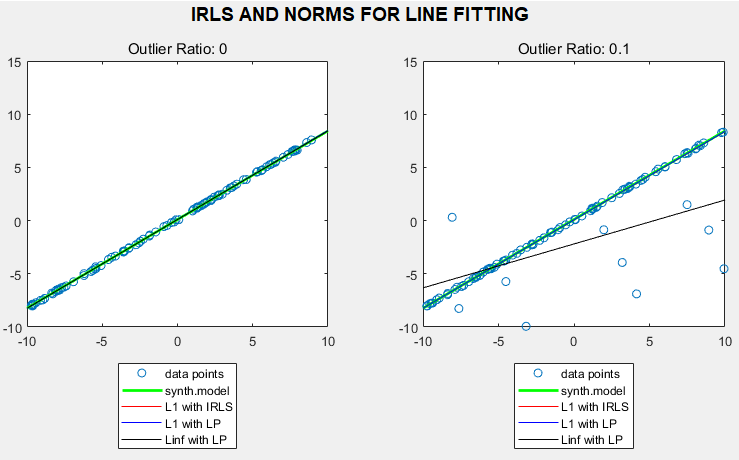


* + Input Value
    - *data*: The data made from *genLineData* function (type: matrix)
    - *Lnorm*: The kind of Lp norm (type: String)
  + Output
    - result\_LP. The result data computed by Linear programming
* **drawLineFittingPlot.m**  
  : This function draws plots of line fitting results(IRLS with and LP with and ) with different outlier ratios. The result is shown in category *Screenshots* below.  
    
  
  + Input Value
    - *a*: The gradient value of the synthesized line
    - *b*: The bias value of the synthesized line
    - *data*: All data made from *genLineData* function (type: cell)
    - *result*: The results of IRLS with made from *doIRLS* function and LP with and made from *doLP* function (type: cell)
    - *outlierRatio*: The ratio of outliers in data for draw outlier data
  + Output
    - No output. It shows result plots.

### **Instructions for running**

1. Open the file “main\_LineFitting.m” in Matlab.
2. Execute that file
   1. The another window that shows plots of IRLS with and LP with  and  norms results is opened.

### **Screenshots**



### **Discuss the results**

In this code, I made synthesized model with outlier ratios 0% and 10%, doing line fitting using  norm (employing IRLS and LP) and  norm (employing LP) with synthesized data, and draw the plots that show the results.

In case of outlier ratio 0%, all methods show similar results each other and also almost match with the synthesized model. In case of outlier ratio 10%, however, the results of norm still similar to the synthesized model but there is quite difference between the result of norm and synthesized model. The purpose of norm is minimize the maximum error among all errors, so it is not robust to outliers. The results of norm with IRLS and LP are almost same but I think LP is more intuitive than IRLS