Exercise 2
Math foundation of computer graphics and vision

Amrollah Seifoddini

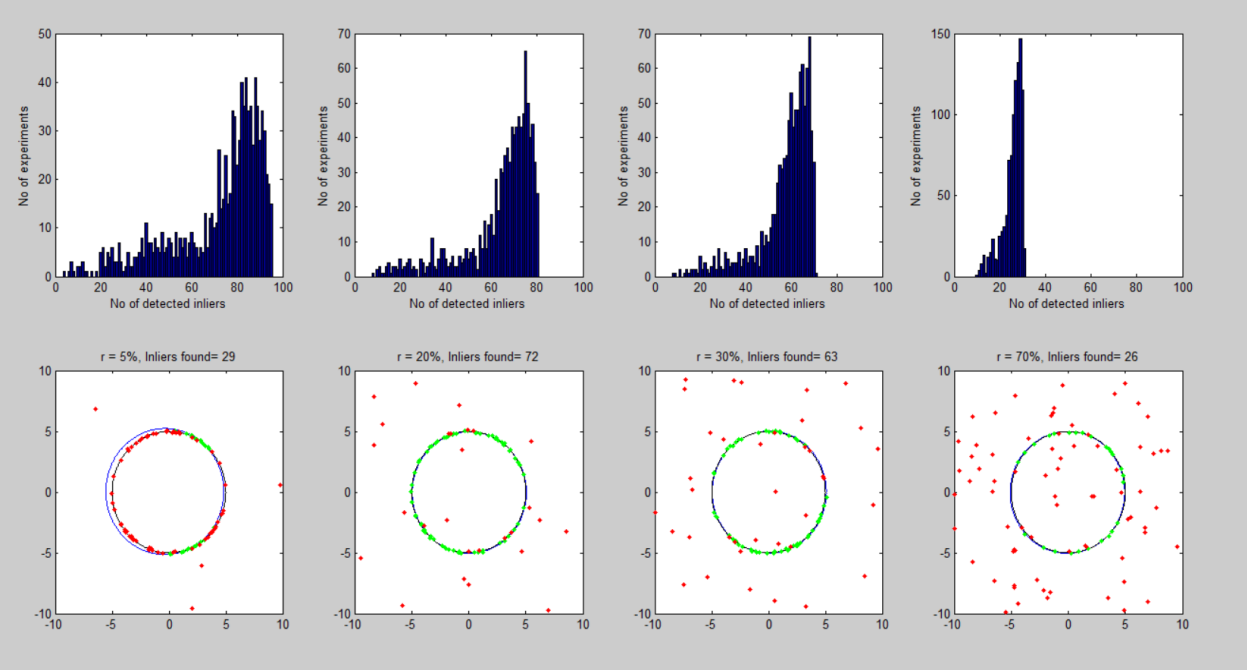
Task 1)

1.1:

Data is generated as described in exercise sheet. And uniform noise between [-.1, .1] is added to inliers.

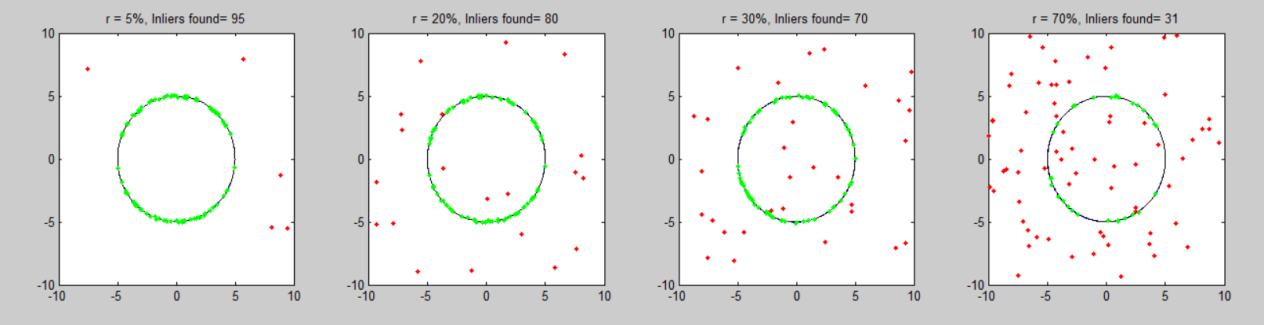
1.2:

Standard RANSAC is implemented for this task. The result of this part for different outlier ratio is displayed in following figure.



1.3:

Exhaustive search implemented for this task. The result circle is shown below.



According to combinatorial math, the number of combinations is n!/((n-k)!k!) where k is 3 for our case and n is 100. So the total number of iterations will be 161700 for 100 data points. But for RANSAC with .99 guarantee and 5 20 30 70% outliers it will be 3, 7, 11, 169 iterations respectively which is much less than exhaustive search.

For 100,000 points, exhaustive search should do 1.6666e+14 iterations. And RANSAC iteration numbers are still the same! So we see a huge advantage in terms of runtime and speed for RANSAC.

Number of inliers found by RANSAC in best solution compared with exhaustive search in table 1.

	5% outlier	20% outlier	30% outlier	70% outlier
RANSAC Inliers #	29	72	63	26
Exhaustive inliers#	95	80	70	31

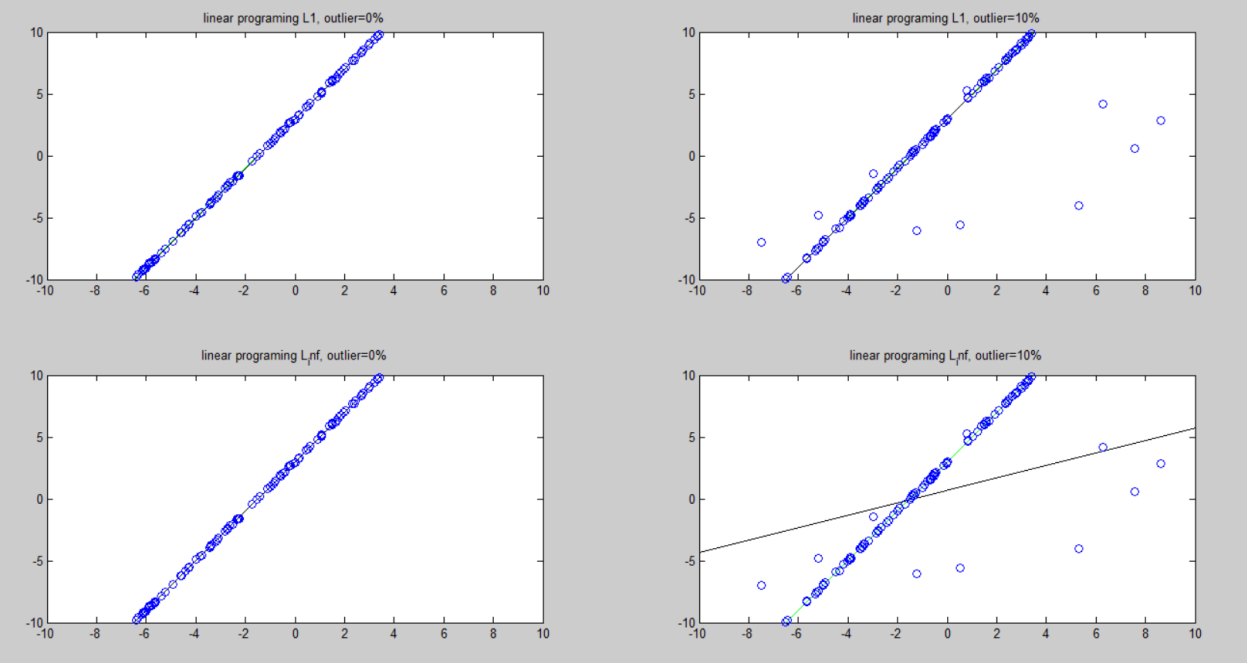
This shows that RANSAC finds less inliers and maybe its solution is not the best solution possible. But it's good enough that we are usually satisfied with the result. And because of its super-fast runtime, we mostly go with RANSAC when we want to do estimation of parameters in presence of noise and outliers for large number of data.

Task 2)

For this task, I used the same approach for data generation for synthesized line. Then I added noise and outliers.

The results for Linear programming with L1 and L_inf are shown below for 0 and 10% outlier ratios.

We clearly see that L_inf is not robust against outliers and will result in a bad fit for 10% outliers. However, L1 is working fine with this much of outlier. I expect that IRLS can also handle bigger outlier ratios which L1 might not do well.


linear programing L1, outlier=0%
linear programing L1, outlier=10%
linear programing L∞f, outlier=0%
linear programing L∞f, outlier=10%