TOPICS IN VIRTUAL REALITY: MATHEMATICAL METHODS FOR VISUAL COMPUTING

EXERCISE 1 - ROBUST ESTIMATION AND NORMS

Handout date: 13 September 2017
Submission deadline: 27 September, 17:59
Demo date: TBA

General Rules

Plagiarism note. Copying code (either from other students or from external sources) is strictly prohibited! We will be using automatic anti-plagiarism tools, and any violation of this rule will lead to expulsion from the class.

Late submissions will not be accepted, except in case of serious illness or emergency. In that case please notify the teaching assistants.

Software. All exercises of this course use the MATLAB programming language. The MATLAB distribution is available from `http://kftp.kaist.ac.kr/` for KAIST students. See our MATLAB tutorial slides for hints or specific functions that could be useful for your implementation.

What to hand in. Send a .zip file of your solution by email to the TA (as a download link, not in attachment). The emails will be processed automatically. The *zip file* must be called "MathMethods17-Ex*-firstname-familyname.zip" (replace * with the assignment number, e.g. MathMethods17-Ex01-John-Smith.zip). The *email subject* must be the same (except ".zip") (e.g. MathMethods17-Ex01-John-Smith). The .zip file MUST contain *a single folder* called the same as the email subject (e.g. "MathMethods17-Ex01-John-Smith") with the following data inside:

- A folder named "code" containing your MATLAB code
- A README file (in pdf format) containing a description of what you have implemented and instructions for running it, as well as explanations/comments on your results.
- Screenshots of all your results with associated descriptions in the README file.

Grading. The homeworks count for 70% of your total grade. Your homework submission will be graded according to the quality of the images/results produced by your program, the conformance of your program to the expected behaviour of the assignment, and your understanding of the underlying techniques used in the assignment. The submitted code must produce exactly the same images included in your submission.

To ensure fairness of your grade, you will be asked to briefly present your work to the teaching assistants. Each student will have about 5 minutes to demo their submission and explain in some detail what has been implemented, report potential problems and how they tried to go about solving them, and point the teaching assistants to the code locations where the various key points of the assignments have been implemented. See above for the scheduled demo date for this particular assignment.

GOAL OF THIS EXERCISE

In this exercise you will apply what you learned about robust optimization, especially RANSAC and Iteratively Reweighted Least Squares (IRLS). You will implement, in Matlab, RANSAC for circle fitting in the presence of outliers, IRLS for line fitting with L_1 norm, and LP for line fitting with L_1 and L_∞ norms.

1. EXERCISE PART 1: RANSAC FOR CIRCLE FITTING

As described in the lectures, RANSAC can be used to fit a model in the presence of outliers and identify these outliers. In this exercise, we will utilize RANSAC for circle fitting with different ratios of outliers.

Given a set of N 2D data points corrupted by outliers, the goal is to detect the dominant circle, that is the circle passing through the highest number of points, up to an inlier threshold. You will create results similar to Figure 1. For this, you will perform the following tasks:

- generate synthetic data sets
- implement and run RANSAC
- plot the distribution of the number of inliers
- implement and run exhaustive search

In this exercise, we consider circle model of the form $(x - x_c)^2 + (y - y_c)^2 = R^2$ where (x_c, y_c) is the circle center and R is the circle radius. The distance between the data points and the circle is defined as the geometrical distance.

1.1. **Data generation.** You will generate a set of N 2D data points on a circle in the domain $[-10, 10] \times [-10, 10]$. These points are corrupted by (small) noise. For this, add random noise (between -0.1 and 0.1) on the (x, y) coordinates of the data points.

Additionally, the data also has outliers, i.e. points that do not "agree" with the circle model. Generate a ratio r of outliers with $r = 5\%$, 20%, 30% and 70%. Define $\tau = 0.1$ as the inlier distance threshold. The total number of points is always N (N=100). If the ratio of outliers is $r = 10\%$, then 10 points are outliers and 90 points are inliers. If $r = 30\%$, then 30 outliers and 70 inliers. When generating the data points, make sure that the synthesized inliers are indeed inliers and the synthesized outliers are indeed outliers and then verify the given outlier ratio r.

1.2. **RANSAC.** You will implement RANSAC as described in the lecture. Compute the number of RANSAC iterations with the generated outlier ratio r, the minimal number of points and a guaranteed accuracy of 99%. Plot the best circle.

1.3. **Exhaustive search.** You will implement exhaustive search on the data points. For this, you will try all the combinations of minimal data points.

A few hints about the implementation:

- you can use the Matlab function nchoosek to get the list of combinations.

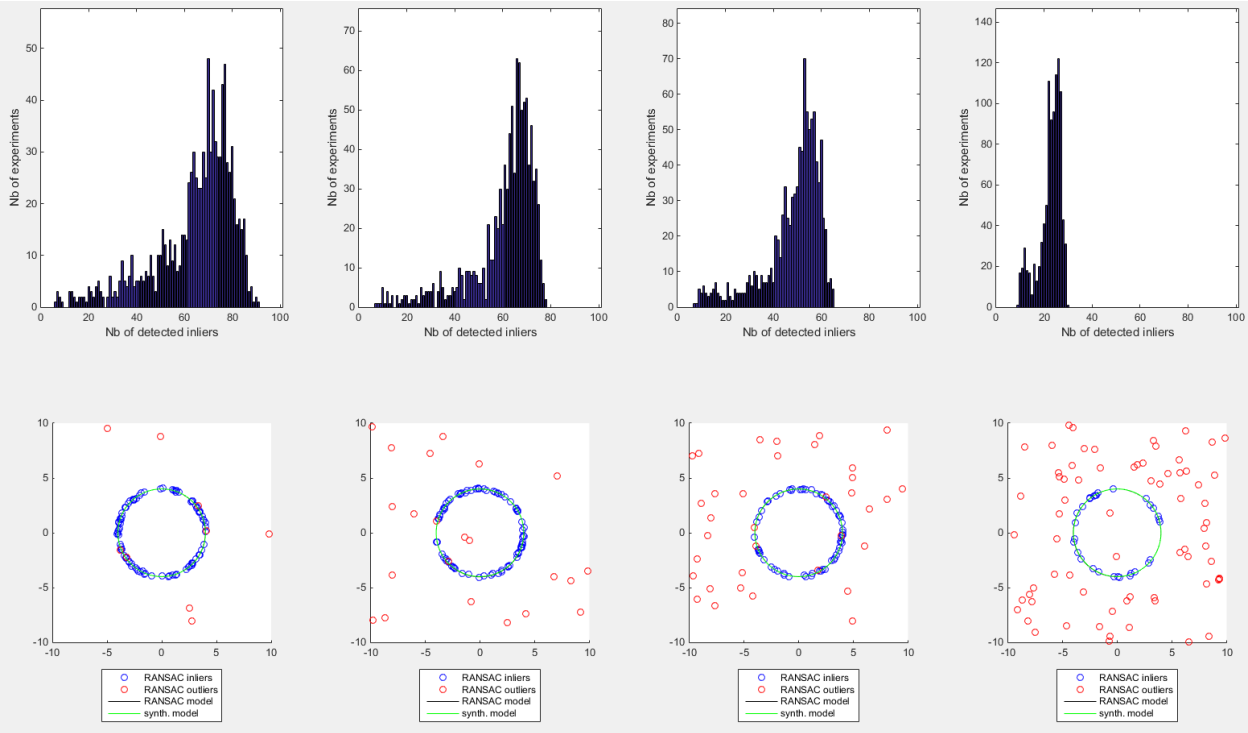

FIGURE 1. Distribution of RANSAC results with different outlier ratios (5%, 20%, 30%, 70%).

- don't forget to initialize the random seed for the *rand* function.
  You can use: *rng('shuffle','twister');*
- instead of a for loop to generate the points, you might want to use a while loop that executes until your points are correctly generated (i.e. inliers are indeed inliers, and outliers are indeed outliers)
- you might want to generate the inliers and the outliers in different while loops, e.g. first a while loop to generate the inliers, and second another while loop to generate the outliers

REQUIRED OUTPUT OF THIS SECTION:

You will create results similar to Figure 1. Show results of circle fitting obtained for four outlier ratios $r = 5\%, 20\%, 30\%$ and 70%. Display the synthesized circle in green, the best result of RANSAC in black, the inliers and outliers found by RANSAC in blue and red respectively.

Show the distribution of the number of inliers found by RANSAC in a histogram (see Fig. 1). Note: the histogram does not correspond to the number of inliers found at each RANSAC iteration: RANSAC is run over M iterations, and the final result counts as one entry in the histogram. Re-apply RANSAC 1000 times, and populate the histogram (i.e. the histogram has 1000 entries).

- Code that generates synthetic data points on a circle with different outlier ratios, applies RANSAC and exhaustive search, and plot the results as shown in Fig. 1
- Answer and discuss the following questions in your submitted readme pdf file: How many combinations (exhaustive search) exist for $N = 100$ points? What about the number of RANSAC iterations with $r = 5\%$, 20%, 30% and 70%? What about when $N = 10,000$ points?
- Discuss and compare the results obtained by RANSAC and exhaustive search in terms of number of inliers, speed, number of synthesized inliers, etc.

2. EXERCISE PART 2: IRLS AND NORMS FOR LINE FITTING

The second task of this assignment is to write a Matlab program to perform line fitting using the L_1 norm (employing IRLS and Linear Programming) and the L_∞ norm (employing Linear Programming), as described in the lecture.

Write a Matlab program which takes as input a set of $N = 100$ 2D data points, fit a line by applying IRLS with L_1 norm and by applying Linear Programming with L_1 and L_∞ norms.

To generate data points on a line, follow a similar strategy to Part1. Add noise (between -0.1 and 0.1) on the (x, y) coordinates of the data points and include 0% (none) and 10% of outliers.

We consider line model $y = ax + b$ and algebraic (vertical) cost.

A few hints about the implementation:

- use *linprog* to perform Linear Programming.
- to add a legend in your figure, you can use *legend*.

You will create results similar to Figure 2. Show results of line fitting obtained for the outlier ratios $r = 0\%$ and 10%. In the same figure, plot the input data, the synthetic line in green, the result of your algorithm in black.

REQUIRED OUTPUT OF THIS SECTION:

- Code that generates data points, runs IRLS with L_1 and LP with L_1 and L_∞ norms
- Show results: plot the synthetic data points, the synthetic line, and the result of IRLS and LP, like in Figure 2
- Discuss the results obtained by these methods in your submitted readme pdf file (e.g. robustness).

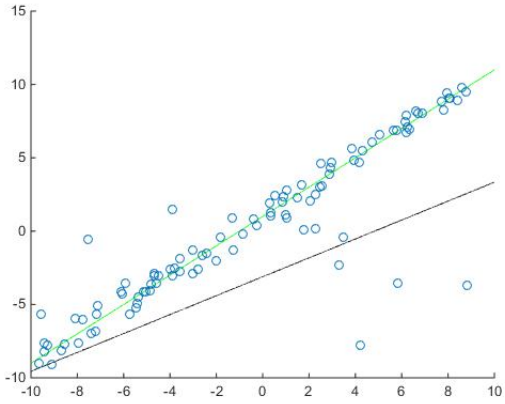

FIGURE 2. Line fitting by Linear Programming example.