Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

MATHEMATICAL FOUNDATIONS OF COMPUTER GRAPHICS AND VISION

EXERCISE 1 - ROBUST ESTIMATION AND OPTIMIZATION

Handout date: 01.03.2022
Submission deadline: 14.03.2022, 23:55

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 assistant and provide a relevant medical certificate.

Software. All exercises of this course use Python. See the exercise session slides and this document for hints or specific functions that could be useful for your implementation.

What to hand in. Upload a .zip file of your solution in moodle. The file must be called "MATHFOUND22-*-firstname-familyname.zip" (replace * with the assignment number). The .zip file MUST contain a single Jupyter notebook "MATHFOUND22-*-firstname-familyname.ipynb" OR a single folder called "MATHFOUND22-*-firstname-familyname" with the following data inside:

- A folder named "code" containing your code
- A README file (in pdf format) containing a description of what you've implemented and instructions for running it, as well as explanations/comments on your results. If using the provided Jupyter notebook, it is sufficient to just comment code your code that is not self explanatory.
- Screenshots of all your results with associated descriptions in the README file (if they are not embedded into a PDF).

Grading. This homework is 8.3% of your final grade. Your submission will be graded according to the quality of the images 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.

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 Python, 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.

You may use the Jupyter notebook that is provided with the exercise. While encouraged, the use of it is optional and you may submit your own Python code or notebook.

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:

(1) generate synthetic data sets
(2) implement and run RANSAC
(3) plot the distribution of the number of inliers
(4) implement and run exhaustive search
(5) implement and visualize exhaustive search on ground truth inliers

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.

Additonally, the data also has outliers - points that don't follow the "circle" rule at all. 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 iterations with the generated outlier ratio r, the minimal number of points and a guaranteed accuracy of 99%. Plot the best circle.

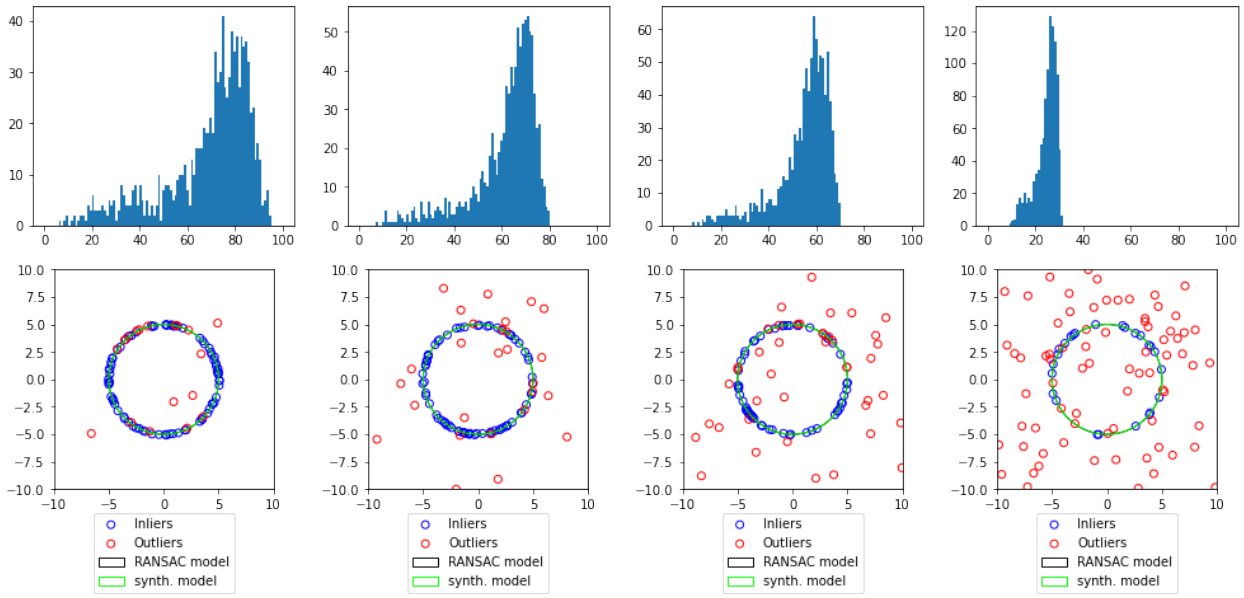

FIGURE 1. Distribution of RANSAC results with different outlier ratios (5%, 20%, 30%, 70%).

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

Additionally, implement an exhaustive search that only considers the ground truth inliers for circle fitting but evaluates on all the points. Show the found inlier histogram of each combination for a single sample with outlier ratio $r = 70\%$. Compare that with the number of inliers found by RANSAC on the same sample by applying RANSAC 1000 times.

A few hints about the implementation:

- We assume you already know basic Python programming. If you need a small refresher, check at the following tutorial[1].
- Use `numpy` to create and manipulate arrays. If you need a small introduction tutorial on numpy, check this link[2].
- You can import `itertools` and use `itertools.combinations` to get the list of combinations.
- If you need to initialize the seed for the `rand` function, import `datetime` and then use `np.random.seed(datetime.now())`.

[1] `https://docs.python.org/3/tutorial/`

[2] `https://numpy.org/doc/stable/user/quickstart.html`

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.

- 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
- Code that plots histogram of exhaustive search on synthetic inliers, and RANSAC on the same sample with outlier ratio $r = 70\%$
- Answer and discuss the following questions:
  - 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 $N = 100,000$ points?
  - Does exhaustive search on all the combinations of data points guarantee the optimal solution (in terms of number of inliers)? Why?
  - Does RANSAC always find close to the optimal number of inliers? Why? If not, would increasing the number of RANSAC iterations improve that?
- 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 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 program which takes as input a set of $N = 100$ 2D data points and fits the 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.

You will create results similar to Figure 2. Show results of line fitting obtained for the outlier ratios $r = 0\%$, 1%, 10%, 50%, and 80%. In the same figure, plot the input data, the synthetic line in green, the result of your algorithm in black.

A few hints about the implementation:

- use `linprog` from `SciPy` to perform Linear Programming (`from scipy.optimize import linprog`).

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.

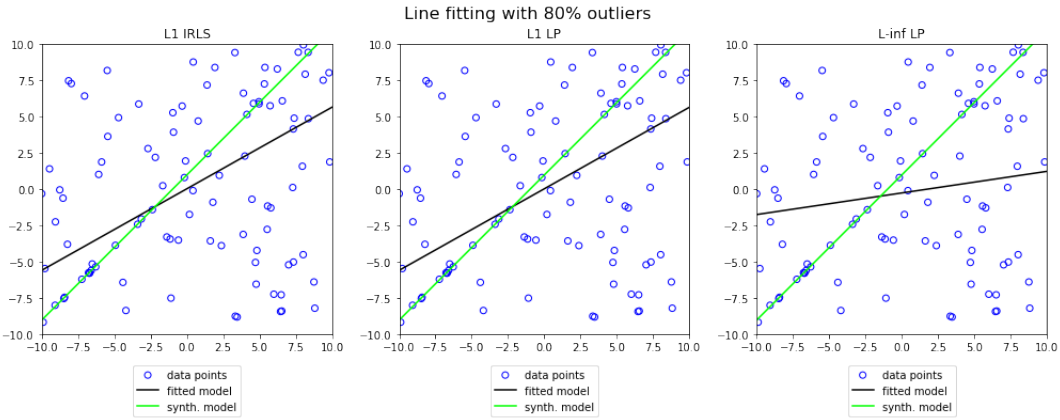

FIGURE 2. Line fitting with $r = 80\%$.