ETH Zürich
Computer Science Department

Prof. Andreas Geiger
Prof. Luc Van Gool
Prof. Vittorio Ferrari
HS 2018

# Computer Vision

# Exercise 4

**Objective:**

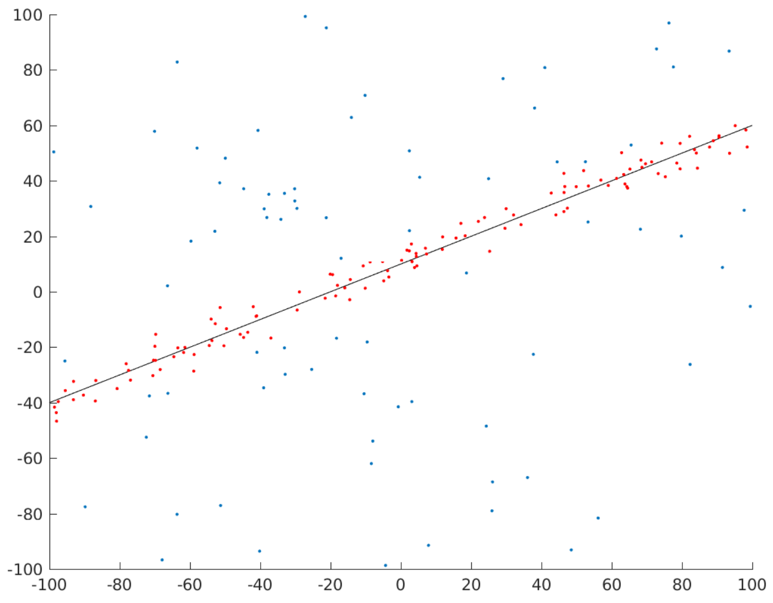
In this exercise you will learn to fit a simple model using RANSAC. Furthermore, in this assignment you will learn to estimate the epipolar geometry between two related views, both manually and automatically by applying RANSAC.

**Documentation:**

Inside the zip archive, the slides can be found in the `doc/` folder. In addition, there is a paper and a book chapter from Hartley and Zisserman, it will complement the information provided. Please take a look in the **`README.txt`** files.

### 4.1 Line fitting with RANSAC (20%)

Implement a line fitting algorithm using RANSAC. A set of points with outliers is generated in the `main_ransacLine.m`. Compare with the real model and least square fitting. Make sure to report the error measured on the true inliers (`err_ls`, `err_ransac` and `err_real` in the code).



## 4.2 Fundamental matrix (25%)

Inside the zip archive there are some image pairs. You will use these images to compute the epipolar geometry between each pair.

Compute the fundamental matrix $F$ that relates two images by implementing the eight-point algorithm. Make sure you include the following in your implementation:

- Your eight-point algorithm should be a homogeneous least-squares solution that includes more than 8 point correspondences. Select the point correspondences by clicking manually on the two images.
- Do not forget to normalize the points before applying the eight-point algorithm (Remove mean and scale the points so that their average distance to the origin is $\sqrt{2}$).
- Enforce the singularity constraint. Draw the epipolar lines for both, the non-singular fundamental matrix $F$ and the fundamental matrix $\hat{F}$ with enforced singularity constraint. Also show the epipoles of $\hat{F}$ in both images (if it is visible in the images), which are the right and left null-vectors of the fundamental matrix.

## 4.3 Essential matrix (10%)

The essential matrix can be computed with known camera calibration. Implement another eight-point algorithm to compute the essential matrix, however this time use the provided camera calibration to compute the essential matrix. You may use the same set of point correspondences from the previous section. Make sure you include the following in your implementation:

- Similar to the eight-point algorithm for the fundamental matrix, your eight-point algorithm for the essential matrix should be a homogeneous least-square solution that includes more than 8 point correspondences.
- Make sure that the normalized image coordinates are used.
- The constraint of the first two singular values must be equal and the third one is zero has to be enforced.

## 4.4 Camera matrix (20%)

The camera matrix $P' = [R|t]$ is made up of the relative transformation $(R, t)$ between the second and the first camera. The first camera matrix $P = [I|0]$ is taken to be the origin of the camera coordinate system. $(R, t)$ can be computed from the essential matrix and four possible solutions exist. Only one of these four solutions is correct.

**a)** Implement the code to compute all the four possible solutions for the camera matrix $P'$ from the essential matrix obtained in the previous section. You should normalize $t$ to unit length and make sure that $R$ follows the right hand coordinate system.

**b)** Choose the correct camera matrix $P'$ from the four possible solutions. Hint: This can be done by checking the triangulated 3D points using the triangulation function provided (`linearTriangulation.m`).

### 4.5 Feature extraction and matching (5%)

Using the same image pairs provided for the previous exercises, now you will extract point features and match them across the image pair. Use a SIFT implementation: VLFeat or Lowe's original SIFT. Proceed on to extract and match the SIFT features from the image pair. Visualize the matches by plotting line segments between the matched points. You can either:

- Display just the first image. For each match, draw a line segment from the point in the first image to the coordinates of the second point, but also in the first image. This gives a sense of the motion of the point from one image to the next.
- Display both images side by side. For each match, draw a line segment from the point in the first image to the point in the second image. This shows exactly which features were matched (this code is given in Lowe's implementation).

**Note:** the showMatchedFeatures() function from Matlab can be used here, as well as the supplied `showFeatureMatches()` in the helpers folder.

### 4.6 8-Point RANSAC (Total 20%)

You will observe several wrong matches (outliers) from the previous step. In this step, you shall implement the RANSAC algorithm to automatically remove the incorrect matches. The eight-point algorithm that you have implemented in Exercise 4.2 will be used to generate the model candidates for the RANSAC algorithm.

**a) Simple RANSAC (15%)**

Implement the eight-point RANSAC algorithm and keep the termination criteria fixed at 1000 trials. The error measure that you should implement to determine if a point correspondence is an outlier or not is given by

$$d(\mathbf{x}', \mathtt{F}\mathbf{x}) + d(\mathbf{x}, \mathtt{F}^\top \mathbf{x}') \tag{1}$$

This computes the sum of the perpendicular distances of the point to the epipolar line in the two respective images. You should specify a threshold, for example 2 pixels, in the RANSAC algorithm for the error measure to determine whether a point correspondence is an outlier or not. Visualize the matches (inliers) from the consensus set with the highest RANSAC score (same visualization as in 4.1). Vary the error threshold and compare the total inlier counts.

**b) Adaptive RANSAC (5%)**

The best solution from RANSAC is given by running an exhaustive search of all possible sets of 8 points from the correspondences but this is often computationally infeasible. It is more efficient to terminate the RANSAC process after $M$ trials if we know with a probability $p$ that at least one of the random samples of $N = 8$ points from these $M$ trials is free from outliers. For this to happen, Equation 2 has to be satisfied.

$$p = 1 - (1 - \mathrm{r}^N)^M \tag{2}$$

$r$ is the largest inlier ratio found after every RANSAC trial. Modify your RANSAC code in Part (a) so that it terminates after $M$ trials following the condition from Equation 2. Fix the value of $p$ at 0.99 in your implementation. Report the value of $M$.

## 4.7 Hand in:

Write a short report detailing your results, no need to explain the methods in detail. However, if one of your implemented algorithms did not seem work, explain why. In addition to what is asked in the exercise explanations, be sure to include:

- A figure with the regressed line with outliers/inliers.
- Image pairs with epipolar lines for each of the Fundamental matrix versions (with and without singularity enforced).
- For the Essential Matrix exercise, be sure to compare the obtained matrix and the matrix obtained in Exercise 4.2 by multiplying this one with the calibration matrix directly (instead of multiplying the image coordinates). Also show the epipolar lines for each image pair.
- Once the Camera Matrix is computed, show that the chosen R and t have the 3D points in front of both cameras (use a 3D plot). The helper function `showCameras.m` can be used here.
- Images overlaid with extracted SIFT features.
- Plots showing inliers and outliers.

The report should not exceed 6 pages in total. Submit the report and (working) MATLAB code to Moodle.