Computer Vision

Lab Assignment - Model Fitting

ETH Zurich, Computer Science Department

Objective

In this lab assignment, you will learn how to use RANSAC (RANdom SAmple Consensus) for robust model fitting, first in the simple case of 2D line estimation and then for fundamental matrix estimation using the 8 point algorithm.

1 Line fitting (Total: 30%)

Implement a line fitting algorithm using RANSAC. Compare your results with the real model and least squares fitting. Make sure to report the error measured on the true inliers in your repport (`err_ls`, `err_ransac` and `err_real` in the code).

2 Fundamental matrix estimation (Total: 40%)

Compute the fundamental matrix F for each image pair by implementing the eight-point algorithm. Make sure to take the following into account for your implementation:

- The eight-point algorithm should be a homogeneous least-squares solution that works for 8 or more point correspondences.
- You should normalize the points before applying the eight-point algorithm (subtract the mean and scale the points). The fundamental matrix should then be rescaled for input image coordinates.
- 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), which are the right and left null-vectors of the fundamental matrix.

For the purpose of this question, you will select the correspondences by clicking manually on the two images. Make sure to include the plots in your report.

3 Feature extraction and matching (Total: 5%)

Install VLFeat [2] to extract and match SIFT [1] features for the same image pairs used before. Use the provided `showFeatureMatches` function to plot the obtained matches in your report.

4 Eight-point RANSAC (Total: 25%)

In the previous step, there were some wrong matches (outliers). In this step, you will use RANSAC along with the fundamental matrix estimation implemented at Step 2 to find the best model.

4.1 Simple RANSAC (15%)

Incorporate the eight-point algorithm inside a RANSAC loop and keep the termination criterion fixed at 1000 trials. As error measure, use the Sampson distance given by

$$\mathcal{S}(x, x') = d(x', Fx) + d(x, F^T x') \ . \tag{1}$$

This computes the sum of the perpendicular distances of the point to the epipolar lines 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 inlier or an outlier. Visualize the matches (inliers) from the consensus set with the highest RANSAC score (same visualization as before). Vary the error threshold and compare the total inlier counts / mean Sampson distance of inliers.

4.2 Adaptive RANSAC (10%)

The best solution for RANSAC in this case is obtained after testing all possible hypotheses (eight point subsets). It is generally more efficient to terminate RANSAC after M trials if we know with a probability p that at least one of the random samples of N (8 in this case) points from these M trials is free from outliers. For this to happen, the following equation needs to be satisfied:

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

where r is the largest inlier ratio found so far at every RANSAC iteration. Modify the code used before so that it terminates after M trials when $p = 0.99$. Report the value of M for each image pair.

Hand in

Hand in your commented MATLAB code and write a short report explaining the main steps of your implementation and addressing the questions from above. Make sure to include the essential parts of your code in the report!
Send the report together with your source code via the moodle submission system (do not send it over email).

References

[1] David G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[2] Andrea Vedaldi and Brian Fulkerson. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. `http://www.vlfeat.org/`. 2008.