Computer Vision – 8-Point Algorithm

WS 2024/25
Prof. Dr. Jochen Schmidt

Technische Hochschule Rosenheim

A Python template program for this exercise is provided in the Learning Campus. We are going to do a 3D reconstruction of a scene from an image pair. To keep things simple we use images from the 2014 Middlebury dataset: https://vision.middlebury.edu/stereo/data/scenes2014/
The image pair for the motorcycle with perfect calibration data is already included in the template ZIP-file.

1) Fundamental Matrix Estimation Without Normalization

In a first step, implement the 8-point algorithm (without normalization) in the pre-defined function computeF(). Some comments describing the required steps are included.

If your implementation is correct, the image displayed after running the program should include blue epipolar lines that are almost horizontal. A high resolution image is also saved in the motorcycle folder (F-nonorm.jpg).

The program also prints two rotation matrices and two translation vectors recovered from F (via E). Does any combination seem to be correct? Have a look at the calib.txt file, which contains calibration information. Which rotation matrix and translation vector would you expect as a result?

2) Fundamental Matrix Estimation With Normalization

In his famous 1997 paper "In Defense of the Eight-Point Algorithm" Richard Hartley showed that this algorithm does not perform as poorly as many researchers thought, if you normalize the input points. This is done as follows (for each image separately):

$$s = \frac{\sum_i \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}}{n\sqrt{2}}$$

Where $\bar{x}, \bar{y}$ are the average x- and y- coordinates in the image (the center of gravity), n is the number of detected points in the image.

Given a detected point with coordinates (x_i, y_i), the normalized coordinates are calculated as $\left(\frac{x_i - \bar{x}}{s}, \frac{y_i - \bar{y}}{s}\right)$

a) What does the 3x3 (homogeneous) transformation matrix T look like that performs this transformation?
b) Implement the function computeNormalizationT() that is supposed to calculate T. Remove the comments to computeNormalizedF() in main() to make sure it will be called.
c) Implement the function computeNormalizedF() that calculates the fundamental matrix F from the normalized point coordinates.

3) Reconstruction of Camera Pose and 3D Scene Points

a) The camera movement between left and right image can be recovered from the fundamental matrix F by computing the essential matrix E from F (how?) and decomposing E into rotation matrix R and translation vector t. There are two solutions for both parameters, resulting in a total of four possible camera configurations.
Which configurations are these (draw a sketch)? How can we determine the correct one?

b) We would like to obtain a sparse 3D reconstruction of the scene using triangulation. How would we implement this?