Version: March 26, 2019

Linköping University

Name of student _____

Personal number _____

Date _____

Teaching assistant _____

Optimisation:
Stereo correspondences for fundamental matrix estimation
Lab Exercise 3

1 Introduction

In this exercise we will do a robust and accurate estimation of the fundamental matrix that relates a pair of images. First we will use the RANSAC (RANdom Sample And Consensus) algorithm to make the estimate, then we will refine the solution using non-linear least-squares optimization with a cost function based on the re-projection error, i.e. the 'Gold Standard' algorithm.

The Python programming language will be used for this lab.

1.1 Preparations

Before the exercise you should have read through this exercise guide and completed the home exercises. They are all clearly marked with a pointing finger. It is also recommended that you read sections 11.1, 11.2, 11.4.1, (11.5 optional) and 11.6 in 'R. Hartley and A. Zisserman *Multiple View Geometry*' (this book is available to you electronically through the library system, and is linked to on the course web page). The algorithms presented in this document are excerpts from these sections of this book, and refer to equations therein.

Expected preparation time: 4h

1.2 Initialization

This lab requires Python 3.6+, *SciPy*, and *OpenCV*. On ISY computers you setup the environment by calling

```
$ module add courses/TSBB15
```

After that you can use `python3 script.py` to run a Python script. For interactive work you can optionally use `ipython3`, which allows e.g. tab-completion and syntax highlighting (you should still use `python3` to run any scripts!).

1.2.1 Help code and notes

The helper functions referenced in assignments can be found by importing the `lab3` module. Familiarize yourself with the available functions:

```
>>> import lab3
>>> help(lab3)
```

Remember: Some parts of the code might receive a huge performance boost by using matrix operations, see https://docs.scipy.org/doc/numpy/user/basics.broadcasting.html for how broadcasting in numpy works.

Objective

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}_i'\}$, determine the fundamental matrix $\mathtt{F}$ such that $\mathbf{x}_i'^{\mathsf{T}}\mathtt{F}\mathbf{x}_i = 0$.

Algorithm

(i) **Normalization:** Transform the image coordinates according to $\hat{\mathbf{x}}_i = \mathtt{T}\mathbf{x}_i$ and $\hat{\mathbf{x}}_i' = \mathtt{T}'\mathbf{x}_i'$, where $\mathtt{T}$ and $\mathtt{T}'$ are normalizing transformations consisting of a translation and scaling.

(ii) Find the fundamental matrix $\hat{\mathtt{F}}'$ corresponding to the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}_i'$ by

  (a) **Linear solution:** Determine $\hat{\mathtt{F}}$ from the singular vector corresponding to the smallest singular value of $\hat{\mathtt{A}}$, where $\hat{\mathtt{A}}$ is composed from the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}_i'$ as defined in (11.3).

  (b) **Constraint enforcement:** Replace $\hat{\mathtt{F}}$ by $\hat{\mathtt{F}}'$ such that $\det \hat{\mathtt{F}}' = 0$ using the SVD (see section 11.1.1).

(iii) **Denormalization:** Set $\mathtt{F} = \mathtt{T}'^{\mathsf{T}}\hat{\mathtt{F}}'\mathtt{T}$. Matrix $\mathtt{F}$ is the fundamental matrix corresponding to the original data $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$.

Algorithm 11.1. *The normalized 8-point algorithm for* $\mathtt{F}$.

2 Epipolar constraint and fundamental matrix estimation

For two images of the same static scene, corresponding points are related by the *epipolar constraint*. The epipolar constraint in homogeneous coordinates assumes the form $(x_2\ y_2\ 1)\,\mathbf{F}\,(x_1\ y_1\ 1)^T = 0$, where $\mathbf{F}$ is the so called *fundamental matrix*, and (x_1, y_1) and (x_2, y_2) is a pair of corresponding points in the two images. In this exercise, you will use the epipolar constraint to simultaneously estimate the fundamental matrix, and a list of corresponding points.

Preparation Question: Read through algorithm 11.1. How might you obtain the input image point correspondences? Why are $n \geq 8$ correspondences needed?

You will not be implementing algorithm 11.1. It is provided for you in a function called `fmatrix_stls`. A number of other routines are made available to you, listed in the following table. All of these functions are useful, so make sure you read their help texts, and understand how they work.

Misc	`harris`	Harris interest point detector.
	`non_max_suppression`	2D non-max-suppression of a feature image.
	`joint_min`	Find an index that is minimal along both rows and columns in a matrix. (e.g. solves the assignment problem)
	`fmatrix_stls`	Estimate the fundamental matrix from at least 8 point correspondences, using the normalised-eight-point-algorithm.
	`cut_out_rois`	Cut out regions of interest (ROIs) and store their pixel values in columns of a matrix.
	`fmatrix_residuals`	Compute the residuals of a set of points w.r.t. the fundamental matrix $\mathbf{F}$. i.e. the signed distances to the epipolar lines in both images.
	`fmatrix_residuals_gs`	Compute the image plane residuals needed for the Gold Standard algorithm.
	`fmatrix_cameras`	Computes two possible cameras from the fundamental matrix $\mathbf{F}$.
	`triangulate_optimal`	Computes optimal 3D triangulation.
	`fmatrix_from_cameras`	Combine camera projection matrices $\mathbf{C}_1$,$\mathbf{C}_2$ into a fundamental matrix $\mathbf{F}$.
Visualisation	`plot_eplines`	Plot epipolar lines given a list of points in the other image, and the fundamental matrix that relates the two images.
	`show_corresp`	Visualise correspondences by drawing lines between corresponding points in image1 and image2.

3 Input data

Wide baseline stereo image pair. One pair is provided, but you may also look for stereo pairs on the web.

Preparation Question: Will the methods work for all stereo image pairs? If not, do you know what cases might cause failures? Consider both camera motion and the geometry of the world.

4 RANSAC

You will now implement a variant of algorithm 11.4. The three principal differences are: (1) you will use the normalized 8-point algorithm at step (iii) (a), instead of the 7 point method. So you will therefore never have three real solutions at step (iii) (d). (2) You are not required to implement steps (iv) or (v). (3) You may select N (number of RANSAC iterations) manually, rather than adaptively.

Preparation Question: In this algorithm, putative correspondences are obtained by similarity of patch intensity (do not use proximity as suggested in algorithm 11.4). What other approach might you use to obtain point correspondences? What would be the advantages/disadvantages?

Objective Compute the fundamental matrix between two images.

Algorithm

(i) **Interest points:** Compute interest points in each image.
(ii) **Putative correspondences:** Compute a set of interest point matches based on proximity and similarity of their intensity neighbourhood.
(iii) **RANSAC robust estimation:** Repeat for N samples, where N is determined adaptively as in algorithm 4.5(p121):
  (a) Select a random sample of 7 correspondences and compute the fundamental matrix F as described in section 11.1.2. There will be one or three real solutions.
  (b) Calculate the distance $d_\perp$ for each putative correspondence.
  (c) Compute the number of inliers consistent with F by the number of correspondences for which $d_\perp < t$ pixels.
  (d) If there are three real solutions for F the number of inliers is computed for each solution, and the solution with most inliers retained.

  Choose the F with the largest number of inliers. In the case of ties choose the solution that has the lowest standard deviation of inliers.
(iv) **Non-linear estimation:** re-estimate F from all correspondences classified as inliers by minimizing a cost function, e.g. (11.6), using the Levenberg–Marquardt algorithm of section A6.2(p600).
(v) **Guided matching:** Further interest point correspondences are now determined using the estimated F to define a search strip about the epipolar line.

The last two steps can be iterated until the number of correspondences is stable.

Algorithm 11.4. *Algorithm to automatically estimate the fundamental matrix between two images using RANSAC.*

Preparation Question: Examine the functions `fmatrix_residuals` and `fmatrix_residuals_gs`. What do these functions compute? What is the difference between the two?

Question: Run your algorithm on an image pair, then use the visualization routines provided. Are the results as expected? Where do the epipolar lines converge? Why? Is the result consistent between runs?

Question: What value on the inlier threshold do you use? Compute the inlier ratio - what is it?

Question: How many RANSAC iterations do you use?

Objective

Given $n \geq 8$ image point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the Maximum Likelihood estimate $\hat{\mathtt{F}}$ of the fundamental matrix.

The MLE involves also solving for a set of subsidiary point correspondences $\{\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i\}$, such that $\hat{\mathbf{x}}'^{\mathsf{T}}_i \hat{\mathtt{F}} \hat{\mathbf{x}}_i = 0$, and which minimizes

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2.$$

Algorithm

(i) Compute an initial rank 2 estimate of $\hat{\mathtt{F}}$ using a linear algorithm such as algorithm 11.1.

(ii) Compute an initial estimate of the subsidiary variables $\{\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i\}$ as follows:

  (a) Choose camera matrices $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$ and $\mathtt{P}' = [[\mathbf{e}']_\times \hat{\mathtt{F}} \mid \mathbf{e}']$, where $\mathbf{e}'$ is obtained from $\hat{\mathtt{F}}$.

  (b) From the correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ and $\hat{\mathtt{F}}$ determine an estimate of $\widehat{\mathbf{X}}_i$ using the triangulation method of chapter 12.

  (c) The correspondence consistent with $\hat{\mathtt{F}}$ is obtained as $\hat{\mathbf{x}}_i = \mathtt{P}\widehat{\mathbf{X}}_i$, $\hat{\mathbf{x}}'_i = \mathtt{P}'\widehat{\mathbf{X}}_i$.

(iii) Minimize the cost

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2$$

over $\hat{\mathtt{F}}$ and $\widehat{\mathbf{X}}_i$, $i = 1, \ldots, n$. The cost is minimized using the Levenberg–Marquardt algorithm over $3n + 12$ variables: $3n$ for the n 3D points $\widehat{\mathbf{X}}_i$, and 12 for the camera matrix $\mathtt{P}' = [\mathtt{M} \mid \mathbf{t}]$, with $\hat{\mathtt{F}} = [\mathbf{t}]_\times \mathtt{M}$, and $\hat{\mathbf{x}}_i = \mathtt{P}\widehat{\mathbf{X}}_i$, $\hat{\mathbf{x}}'_i = \mathtt{P}'\widehat{\mathbf{X}}_i$.

Algorithm 11.3. *The Gold Standard algorithm for estimating* $\mathtt{F}$ *from image correspondences.*

5 Non-linear least-squares and the Gold Standard algorithm

You will now implement the Gold Standard algorithm (11.3)[1].

In step (iii) of the algorithm you will use the following function:

- Use the `least_squares` function in *SciPy*

```
>>> from scipy.optimize import least_squares
```

Preparation Question: Refer to the documentation for the above function to ensure that you understand its syntax. You need to provide a starting guess, a residual function, and constant parameters (the observed points). Write out the function call you will use.

Question: Run your algorithm on an image pair then use the visualization routines provided. Are the results as expected? Where do the epipolar lines converge? Why?

[1] see page 581 in Multiple View Geometry for definition of matrix cross product operation used in this algorithm

Question: Comment on the differences (if any) between the results using RANSAC and the Gold Standard algorithm.

6 Verification

Preparation Question: Using the tools (functions) provided with this lab, how will you quantitatively compare the solutions generated by the different algorithms?

Question: Evaluate the algorithms you have implemented. Write down qualitative and quantitative comparisons.

7 Sparsity pattern for the Jacobian

This exercise should be done if there is time left.

Extra

In the Gold Standard Algoritm (and in Bundle Adjustment problems in general) most of optimisation time is spent on updating the system Jacobian, $\mathbf{J}$. The Jacobian contains the partial derivatives of the residual vector $\mathbf{r}$ with respect to the parameter vector $\mathbf{p}$:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial r_1}{\partial p_1} & \cdots & \frac{\partial r_1}{\partial p_N} \\ \vdots & \ddots & \\ \frac{\partial r_K}{\partial p_1} & \cdots & \frac{\partial r_K}{\partial p_N} \end{bmatrix}$$

As most values in the Jacobian will be zero, much time can be saved by telling the optimizer to never compute these. This is done by providing a sparsity mask for the Jacobian:

- Use the `jac_sparsity` parameter to `least_squares`

Question: Examine `fmatrix_residuals_gs` and write down the order of the residuals in the returned residual vector, and the order of the parameters in the parameter vector.

Now write a function that computes a suitable sparsity pattern as a function of the number of point correspondences. Run the optimization without and with your sparsity pattern provided. To measure time

- Use the `time.time()` function (Note: must `import time` first)

Question: Compare the accuracy of the two solutions. Are they the same? What speedup factor did you get?

Note that even more speedups of Bundle Adjustment are possible:

1. By default, the Jacobian is computed by finite differencing. If you derive the analytical expressions for each non-zero element in the Jacobian, the Jacobian can be computed explicitly in the cost-function.
2. If you write the optimizer yourself, you can use the problem specific structure of $\mathbf{J}^T\mathbf{J}$ to simplify the computation of the update step. A common way to do this is to use the *Schur complement*.

8 Better point correspondences

This exercise should be done if there is time left.

Extra

You will now try to use interest point matching to generate a better set of point correspondences. To do this, you will use the SIFT or ORB features:

- Both SIFT and ORB are available in *OpenCV*, and can be accessed by

```
>>> import cv2
>>> sift = cv2.xfeatures2d.SIFT_create()
>>> orb = cv2.ORB_create()
```

  See https://docs.opencv.org/3.4.3/db/d27/tutorial_py_table_of_contents_feature2d.html for an introduction. Note in particular that ORB is a binary feature and should thus be matched using the Hamming distance instead of least squares, or scalar products, as discussed in Lecture 8.

8.1 Questions

Question: Comment on the quality of the point correspondences

Use the correspondences you get from **cv2**, as input to your two algorithms.

Question: What impact does using SIFT (or ORB) point correspondences have on the solutions?