Assignment 2
Due date for part 1: November 24, 2020
Due date for part 2: December 8, 2020

Computer Vision
University of Bern
Fall 2020

Introduction

In this assignment you will apply the theory on 3D reconstruction from the Computer Vision class and implement basic algorithms for epipolar geometry. The objectives are:

1. To estimate the **fundamental matrix** and **epipolar lines**;
2. To reconstruct a 3D model from two views using the essential matrix.

Required work

- Implement your solutions in Python in a Jupyter notebook.
- Write a report (pdf format) describing the task, your solution and the results you have obtained.

Submission details

- **Important**: No group work allowed – it must be only your own coding and writing.
- The due date of the submission of the first part of the assignment is **November 24, 2020** and for the second part of the assignment is **December 8, 2020**.
- Submit your work in **ILIAS**.
- Submit the report (only for part 2) in one **PDF** file.
  The report will be accepted only in **PDF** format!
- Submit your commented **Python code** as a zip file (1 file for part 1 and 1 file for part 2).
  - **Important**: Submit your Jupyter notebooks with all outputs displayed. To obtain credits, the code must run without crashing or producing errors and warnings (thoroughly test your code by running it on a different computer before submitting it).
  - Code errors will incur a penalty in the final assignment mark.

Description

This assignment is made of two parts. In the first part, you need to estimate the fundamental matrix using the (normalized) Eight Point Algorithm and compute epipoles and epipolar lines. You will test your implementation on our sample data. For the second part, we provide a synthetically rendered pair of image coordinates (from correspondences across two views of a cow) and ask you to estimate the camera poses and reconstruct the 3D geometry.

Fig. 1: A random subset of matched points between two views.

Part 1: The Eight Point Algorithm (due on November 24, 2020)

[Total 50 Points]

First, carefully read the Jupyter notebook(s) that we provide. The cells marked with TODO and corresponding functions in `utils.py` are to be completed with your implementation. You will then submit your updated Jupyter notebook.

1. Complete the function `get_normalization_matrix` that normalizes the 2D points as described on slide 33 (lecture 8). **[10 Points]**

2. Complete the function `eight_points_algorithm` that estimates the fundamental matrix relating the two views and display the estimated matrix. Make sure that the estimated fundamental matrix has a rank of 2 (add code to verify this property). For the fundamental matrix and the relationship between corresponding points please refer to the slides 23 - 35 (lecture 8). **[25 Points]**

3. Complete the functions `right_epipole` and `plot_epipolar_line` and visualize their outputs. Derive a formula to plot the epipolar lines. In the comments section next to your code explain the meaning of epipolar lines and epipoles. **[15 Points]**

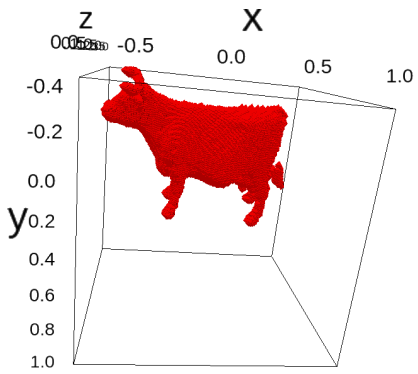

Fig. 2: Reconstructed 3D points of a cow.

Part 2: 3D Model Reconstruction (due on December 8, 2020)

[Total 50 Points]

1. You are given a new set of 2D point correspondences. Some of these points are outliers due to poor feature matching. Which points are outliers is unknown to you and you need to detect them by using the Random Sample Consensus algorithm (RANSAC). Implement your own RANSAC algorithm by following slide 22 (lecture 7). At every iteration, RANSAC requires you to select a subset of 2D points, to compute the fundamental matrix only on the selected set (use the code for Part 1 – solutions will also be provided right after your submission of Part 1), and to compute the re-projection error on the 2D points. The error will allow to count the inliers (use a reasonable threshold). After a fixed number of iterations, choose the fundamental matrix estimate that gives the highest number of inliers. Then, based on this estimate compute the re-projection error on all 2D points and detect the outliers. **[10 Points]**

2. Using the known intrinsic parameters (provided in the Jupyter notebook) compute the essential matrix. Please refer to the slide 27 (lecture 8). **[10 Points]**

3. Complete the function `decompose_essential_matrix` that estimates up to an unknown sign the rotation and translation between the views and display them. **[20 Points]**

4. Use the function `infer_3d` to reconstruct the 3D points using the obtained rotation and translation and choose a consistent solution. **[10 Points]**

   In the Jupyter notebook:

   (a) Show the reconstructed 3D model.

   (b) Show the errors of the estimated extrinsics of the cameras. Use the ℓ^1-norm to measure the difference between estimated and ground-truth translations. Do the same for rotation by e.g. converting the matrix to Euler angles. Note that the true extrinsic parameters are provided so that you can verify your computations.

See the file `Intrinsic and extrinsic parameters.rtf`. Note that the poses in this file are absolute. If you want to compare them to your estimates, you need to convert them into the relative reference frame.

Note: The corresponding points are described in a text file. The first line in this file contains the number of matched points. Subsequent lines contain the description of pairs of corresponding points. Each pair of corresponding points is described by four coordinates: x_r, y_r, x_l, y_l, where (x_r, y_r) and (x_l, y_l) are corresponding points.