ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision: Assignment 6
Exam Prep Session

December 7, 2023

Department of Computer Science, ETH Zürich

Chapter 1

Introduction

1.1 Environment Setup

For setting up the Python environment, we will use miniconda. For the best and easiest experience, we strongly recommend using a Linux distribution (such as Ubuntu). However, miniconda is also compatible with Windows and Mac.

Run the following commands from the root of the code directory to install and activate the environment.

```
conda env create -f env.yaml
```

```
conda activate cv-lab6
```

1.2 Hand in

The deadline for this assignment is December 22, 23:59. Write a short report answering the questions mentioned in the tasks. Upload it together with your code and images in a single zip file to moodle.

Throughout the assignment, do not modify the function interfaces or the already written complete code. We have added extra comments to the base code to guide you. We use `TODO` to mark the parts that you need to fill in in the codes.

In case you run into any issues, please check the moodle forum and open a new topic if needed.

Chapter 2

Structure from Motion

In this assignment, you will produce a reconstruction of a small scene using Structure from Motion (SfM) methods.

We combine realtive pose estimation, absolute pose estimation and point triangulation to a simplified Structure from Motion pipeline. The code framework for this is provided in the file `sfm.py`. We provide you with extracted keypoints and features matches. These matches are already geometrically verified and do not contain outliers since we would need extra steps to handle outliers in the pipeline. Before you start, use the visualization in the framework to verify that the data is loaded correctly and to get an idea of the scene. This can help you judge if your results are correct later on. You can comment out the visualization after this to save time when testing your code. To initialize the scene we will need to estimate the relative pose between two images. Afterwards we can triangulate the first points in the scene and then iteratively register more images and triangulate new points.

2.1 Essential matrix estimation (10 pts.)

We provide the camera intrinsics matrix K, so we can compute the relative pose between the first images from the essential matrix E. We use DLT as in the calibration exercise to reformulate the constraint

$$\hat{x}_1^T E \hat{x}_2 = 0. \tag{2.1}$$

Remember to lift the keypoints to the **normalized** image plane first, i.e., $\hat{x} = K^{-1}x$. When you obtained an estimate from DLT you need it to fulfill the essential matrix constraints:

- the first two singular values need to be equal (and > 0)
- the third singular value is 0

You can do this by using SVD on your estimated essential matrix and set the singular values accordingly, i.e., make the first two singular values to be equal and the third singular value as 0. Since E is up to scale, you can just set the first two singular values to 1.

Fill in function `EstimateEssentialMatrix` in `geometry.py`.

2.2 Point triangulation (5 pts.)

The DLT for point triangulation is already implemented in the framework (`TriangulatePoints` in `geometry.py`).

However, some points that fulfill the mathematical constraints may lie behind one or both cameras. Since these points are clearly wrong, implement a filtering step in the point triangulation. Fill in the remaining part of `TriangulatePoints`.

2.3 Finding the correct decomposition (10 pts.)

The decomposition of the essential matrix into a relative pose is already implemented (`DecomposeEssentialMatrix` in `geometry.py`).

However, this gives four different poses that all fulfill the requirements of the essential matrix. To find the actually correct one, try to triangulate points with each one and use the one with the most points in front of both cameras. Fill in the corresponding part of `sfm.py`. *Hint: Set one of the camera poses to identity and be careful with the direction of the transformation.*

2.4 Absolute pose estimation (2 pts.)

Given a point cloud and correspondences to the image keypoints we can estimate the absolute pose of a new image with respect to the scene. Fill in the missing part of `EstimateImagePose` in `geometry.py`.

2.5 Map extension (10 pts.)

Once we know the new images pose we can extend the map by triangulating new points from all possible pairs of registered images. Implement the triangulation with all pairs and make sure to correctly add the new 2D-3D correspondences to each image. Fill in the `TriangulateImage` of `geometry.py` and `UpdateReconstructionState` of `corrs.py`.

2.6 Result (3 pts.)

Save the plot and add it in the report.

Chapter 3

Model Fitting

In this assignment, you will learn how to use RANSAC (RANdom SAmple Consensus) for robust model fitting. We will work on the simple case of 2D line estimation.

3.1 Line Fitting

With a ground truth linear model $y = kx + b$, we generate a point set `(x_noisy, y_noisy)` with the linear model and add noise to it. Besides, we also add outliers to the point set.

3.1.1 Least-squares Solution (3 pts.)

Fill in function `least_square` in `line_fitting.py`. The function returns the least-squares soultion, `k_ls` and `b_ls`, given x and y. (*hint*: you can use `numpy.linalg.lstsq`).

3.1.2 RANSAC (7 pts.)

Fill in function `ransac` in `line_fitting.py`. The function (loops for `iter=300` iterations) works as follows:

1. randomly choose a small subset, with `num_subset` elements, from the noisy point set (*hint*: you can use `random.sample` to choose a set of random indices)
2. compute the least-squares solution for this subset
3. compute the number of inliers and the mask denotes the indices of inliers (a point is an inlier if its distance to the line smaller than `thres_dist`), fill in function `num_inlier`

4. if the number of inliers is larger than the current best result, update the parameters `k_ransac`, `b_ransac` and also the `inlier_mask`

3.1.3 Results (5 pts.)

For k, b, **write down the ground truth, estimation from least-squares and estimation from RANSAC in the report**.

Save the plot and add it in the report.