Computer Vision 2020
HW#1
Due on February 4, 2020

Submission:

- *Please use your programming language of choice, MATLAB, C, python, …*
- *Please prepare a .pdf report file containing your answers.*
- *Compress your report file and your codes into a single zip file.*
- *Name the file as your last name followed by hw1 for example: shirani_hw1.zip*
- *Submit by uploading to Avenue (under Assessment/Assignment). Please do NOT email me your file.*

1. Camera Calibration

Camera calibration is one of the most fundamental vision problems. It refers to the process of calculating the intrinsic and extrinsic parameters of the camera, which are necessary for many applications such as 3D scene reconstruction. The goal of this problem is to implement a calibration algorithm based on the method described in the class. Provided an input image, we want to compute the intrinsic (focal length, principal point, etc.) and extrinsic (rotation and translation) parameters of the camera used to capture this image.
A typical way to calibrate a camera is to take a picture of a calibration object, find 2D features in the image and derive the calibration from the 2D features and their corresponding positions in 3D. In our case, we use a cube as a calibration object, textured with a checkerboard pattern. We detect in the image the 2D features corresponding to the corners of each tile on the checkerboard.

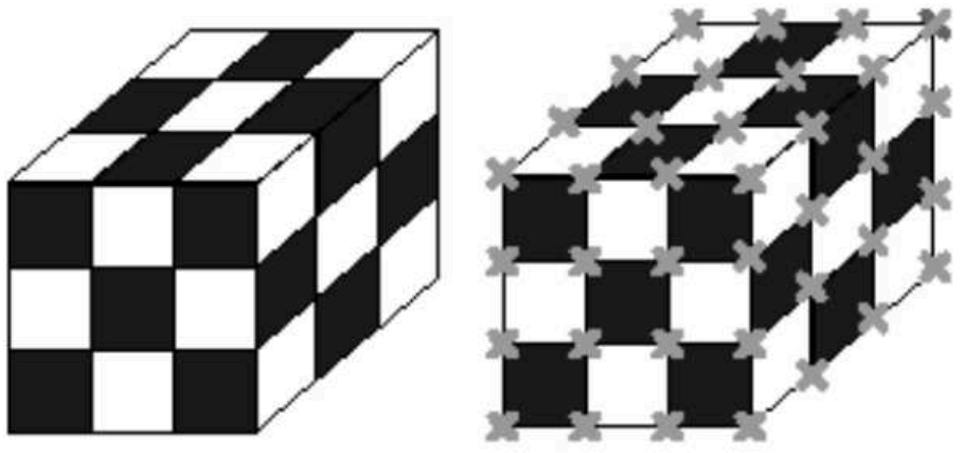

Figure 1: Input image and detected features. On the left is the original image and on the right are the detected feature points.

Since we know its size, we can find the 3D position of each 2D feature relative to the center of the cube. Files Feature2D.txt and Feature3D.txt contain the 2D corner features and the corresponding 3D positions.

1.1 Linear camera calibration

Calibrate a projective camera using a simple linear least squares approach. Given a data file that contains 3D coordinates of some points in the scene, along with their corresponding 2D projections in the image, you are to write a function called LinearCalib that computes the projective camera parameters. The signature of the function should be as follows:

function [CamMatrix] = LinearCalib(Points3D, Points2D)
Input:
Points2D = a (2 x N) matrix of N 2D points
Points3D = a (3 x N) matrix of N 3D coordinates
Output:
CamMatrix = 3 x 4 projective camera matrix

One way to solve this problem is using the "direct linear transformation" algorithm in which you set up the problem as solving a linear system of the form Ax=0 where x is a 12x1 column vector of the entries for the 3 x 4 projective camera matrix and A is a 2n x 12 matrix as described in the lecture. Compute the SVD of A and then the solution is the unit eigenvector with the least eigenvalue. Be sure to describe the steps of the method you used in your writeup. Decompose the projection matrix P into KR [I | -t] and compute the resulting intrinsic and extrinsic parameters.

1.2 3D to 2D projection

You can check the accuracy of your camera calibration result by projecting the given 3D points (in homogeneous coordinates) using the camera matrix that was obtained by your linear camera calibration method. Write a function:

function [ProjPoints2D] = CameraProject(Points3D,CamMatrix)
Input:
Points3D = a (3 x N) matrix of N 3D coordinates
CamMatrix = 3 x 4 camera matrix
Output:
ProjPoints2D = a (2 x N) matrix of N 2D points

Make sure that the points computed by ProjPoints2D are close to the given Points2D matrix to ensure correctness.
Transform the three 3D scene points: (0, 0, 0), (1, 1, 1), (5, 5, 5) and three other points chosen by you onto the image plane. Comment on any errors that you find.
Hint: make sure you convert to homogenous coordinates. The data in file Feature3D.txt is already in homogenous form.

2. Fundamental Matrix Estimation from Point Correspondences

This programming question is concerned with the estimation of the fundamental matrix (F) from point correspondences using the eight-point algorithm discussed in the class. Find the fundamental matrix F. The F that you find might not be of rank 2 because of measurement errors. So we have to enforce the rank-two constraint for the fundamental matrix. In order to do that use SVD to decompose F to $F=UDV^T$ where D=diag(r,s,t), r>s>t. Find a new F by setting t=0 and $F_{new}=U\,\text{diag}(r,s,0)V^T$.
The data for this assignment can be found in data.zip. The zipped file contains two datasets. For each dataset, you will find in the corresponding directory the files
pt_2d_1.txt
pt_2d_2.txt
image1.jpg
image2.jpg

The first two of these files contain matching image points (in the following format: number n of points followed by n pairs of coordinates, the coordinates are v first and u second). The two remaining files are the images where the points were found.

You must turn in:
2.1 For each data set the calculated F matrix.
2.2 For each data set the average distance between the points and the corresponding epipolar lines for each image, as well as a drawing similar to Figure 2 (with the selected corresponding points and their epipolar lines).

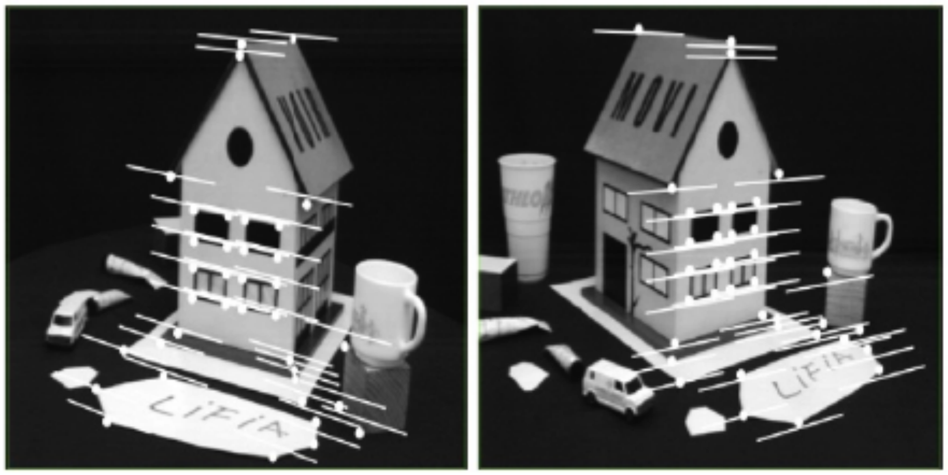

Figure 2: Example illustration, with epipolar lines shown in both images (Image courtesy Forsyth & Ponce)