Don Johnson

CS 585

1 Assignment 12 (100 points)

1.1 Learning Objectives

1.2 Programming Assignment

1.2.1 The Camera Matrices

1. Write down the internal calibration matrix, K, using the internal parameters given above.

**The internal calibration matrix K:**

**1250 0 500**

**0 1250 500**

**0 0 1**

2. Write down the camera matrix, P0 for the first camera

**The augmented identity matrix I3\_Aug used to create P0:**

**1 0 0 0**

**0 1 0 0**

**0 0 1 0**

**The P0 camera matrix = K \* I3\_Aug:**

**1250 0 500 0**

**0 1250 500 0**

**0 0 1 0**

3. Write down the camera matrix, P1, for the second camera

**The rotation matrix Rp1 used to create P1:**

**0.875 0 -0.484123**

**0 1 0**

**0.484123 0 0.875**

**The translation matrix Tp1 used to create P1:**

**4.84123**

**0**

**1.25**

**The P1 camera matrix = K \* [Rp1 | Tp1]:**

**1335.81 0 -167.654 6676.54**

**242.061 1250 437.5 625**

**0.484123 0 0.875 1.25**

4. Use your camera matrices to determine the image coordinates of the fixation point. When your

camera matrices are correct, the fixation point, x0 should project to the image coordinates

[500, 500] in both cameras.

**Image plane point u0\_0 corresponding to the fixation point x0**

**500**

**500**

**Image plane point u0\_1 corresponding to the fixation point x0**

**500**

**500**

1.2.2 Computing Image Coordinates

Below, I have given a collection of 3D world points arranged in a special way. I have chosen a point

3 meters above the fixation point: [0; 3; 10]. This point projects to the image coordinates [500; 875]

in both images. To help you visualize the points, I have provided a top-down drawing. Compute

the image coordinates of the following points for both cameras. In the remainder of the assignment,

the image coordinates will be referred to as un;0 for the first camera and un;1 for the second camera.

**x y z**

**x1** 0 3 10

**x2** 0 2.4253 8.0843

**x3** 0 3.5747 11.9157

**x4** 0.9274 2.4253 8.3238

**x5** -0.9274 3.5747 11.6762

**Image plane point u1\_0 corresponding to world point x1**

**500**

**875**

**Image plane point u1\_1 corresponding to world point x1**

**500**

**875**

**Image plane point u2\_0 corresponding to world point x2**

**500**

**875.002**

**Image plane point u2\_1 corresponding to world point x2**

**639.275**

**864.213**

**Image plane point u3\_0 corresponding to world point x3**

**500**

**874.999**

**Image plane point u3\_1 corresponding to world point x3**

**400.713**

**882.69**

**Image plane point u4\_0 corresponding to world point x4**

**639.269**

**864.212**

**Image plane point u4\_1 corresponding to world point x4**

**725.856**

**837.511**

**Image plane point u5\_0 corresponding to world point x5**

**400.717**

**882.691**

**Image plane point u5\_1 corresponding to world point x5**

**315.869**

**905.563**

1.2.3 Epipolar Lines

The observant reader may notice that the points x1, x2, x3 and t0 are all co-linear, as are the

points x1, x4, x5, and t1. Remember that to compute the coefficients of a 2D line connecting two

points, you can represent the points in homogenous coordinates and take the cross product, leading

to the coefficients [A B C] for the formula Ax + By + C = 0. I recommend normalizing the result

coeffcients so that B = 1 so that you can compare the equations. I have implemented a function

to compute the Fundamental matrix, given two camera matrices.

**Used modified HW12.cpp to calculate previous answers; confirmed with MATLAB. From this point forward, used MATLAB.**

1. Compute the image coordinates of the epipole e1, the image of the camera position of t0 in

the second camera (It will not be inside the image)

**>> P1**

**P1 =**

**1335.8 0 -167.65 6676.5**

**242.06 1250 437.5 625**

**0.48412 0 0.875 1.25**

**>> t0**

**t0 =**

**0 0 0 1**

**>> e1=P1\*t0'**

**e1 =**

**6676.5**

**625**

**1.25**

**>> e1=e1/e1(3)**

**e1 =**

**5341.2**

**500**

**1**

2. Compute the image coordinates of the epipole e0, the image of the camera position t1 in the

first camera. (This will also not be in the image.) The two epipoles will not appear to be

symmetric, but you should take a second to think about the business with the principal point

to see why.

**>> P0**

**P0 =**

**1250 0 500 0**

**0 1250 500 0**

**0 0 1 0**

**>> t1**

**t1 =**

**4.8412 0 1.25 1**

**>> e0=P0\*t1'**

**e0 =**

**6676.5**

**625**

**1.25**

**>> e0=e0/e0(3)**

**e0 =**

**5341.2**

**500**

**1**

3. Compute the coefficients of the line through the image points u2;1 and u3;1 using cross products

**u2\_1 =**

**639.28**

**864.21**

**1**

**>> u2\_1**

**u2\_1 =**

**639.28**

**864.21**

**1**

**>> u3\_1**

**u3\_1 =**

**400.71**

**882.69**

**1**

**>> cross(u2\_1,u3\_1)**

**ans =**

**-18.476**

**-238.56**

**2.1798e+05**

**>> line\_u2\_1\_u3\_1=line\_u2\_1\_u3\_1/line\_u2\_1\_u3\_1(2)**

**Coefficients of line\_u2\_1\_u3\_1 =**

**0.077449**

**1**

**-913.72**

**Confirm the line is correct (sum should equal 0 with available numerical precision):**

**>> line\_u2\_1\_u3\_1=line\_u2\_1\_u3\_1/line\_u2\_1\_u3\_1(2)**

**line\_u2\_1\_u3\_1 =**

**0.077449**

**1**

**-913.72**

**>> line\_u2\_1\_u3\_1(1)\*u2\_1(1)+line\_u2\_1\_u3\_1(2)\*u2\_1(2)+line\_u2\_1\_u3\_1(3)\*u2\_1(3)**

**ans =**

**-1.1369e-13**

**>> line\_u2\_1\_u3\_1(1)\*u3\_1(1)+line\_u2\_1\_u3\_1(2)\*u3\_1(2)+line\_u2\_1\_u3\_1(3)\*u3\_1(3)**

**ans =**

**-1.1369e-13**

4. Compute the coefficients of the line from the epipole e1 and the image points u1;1

**>> e1**

**e1 =**

**5341.2**

**500**

**1**

**>> u1\_1**

**u1\_1 =**

**500**

**875**

**1**

**>> line\_e1\_u1\_1=cross(e1,u1\_1)**

**Coefficients of line\_e1\_u1\_1 =**

**-375**

**-4841.2**

**4.4236e+06**

**>> line\_e1\_u1\_1=line\_e1\_u1\_1/line\_e1\_u1\_1(2)**

**Coefficients of line\_e1\_u1\_1 =**

**0.07746**

**1**

**-913.73**

**Confirm the line is correct (sum should equal 0 with available numerical precision):**

**>> line\_e1\_u1\_1(1)\*e1(1)+line\_e1\_u1\_1(2)\*e1(2)+line\_e1\_u1\_1(3)\*e1(3)**

**ans =**

**-1.1369e-13**

**>> line\_e1\_u1\_1(1)\*u1\_1(1)+line\_e1\_u1\_1(2)\*u1\_1(2)+line\_e1\_u1\_1(3)\*u1\_1(3)**

**ans =**

**-1.1369e-13**

5. The line connecting the epipole e1 and the image points u1;1 is the epipolar line corresponding

to which image points? Hint: The image points are from the first camera, and there are three.

**Substituting each of the six image points into the equation for the line and see if the answer is zero (with available numerical accuracy) gave u1\_0=(500,800), u4\_0=(639.27, 864.21) and u5\_0=(400.72, 882.69).**

**>> line\_e1\_u1\_1(1)\*u1\_0(1)+line\_e1\_u1\_1(2)\*u1\_0(2)+line\_e1\_u1\_1(3)\*u1\_0(3)**

**ans =**

**1.9363e-09**

**>> line\_e1\_u1\_1(1)\*u4\_0(1)+line\_e1\_u1\_1(2)\*u4\_0(2)+line\_e1\_u1\_1(3)\*u4\_0(3)**

**ans =**

**-0.0005864**

**>> u4\_0**

**u4\_0 =**

**639.27**

**864.21**

**1**

**>> line\_e1\_u1\_1(1)\*u5\_0(1)+line\_e1\_u1\_1(2)\*u5\_0(2)+line\_e1\_u1\_1(3)\*u5\_0(3)**

**ans =**

**0.00041804**

6. Write down the Fundamental matrix

**The fundamental matrix computed from P0, P1, and t0:**

**0 -1 500**

**-1 -2.62444e-16 -4341.23**

**500 5341.23 -500000**

7. Use the Fundamental matrix to compute the epipolar line corresponding to u1;0.

**>> epipolar\_line=F\*u1\_0**

**epipolar\_line =**

**-375**

**-4841.2**

**4.4236e+06**

**Since the epipolar line for u1\_0 is the line connecting this point and the epipole in the same image plane, confirm these coefficients A,B and C for the epipolar line with:**

**>> cross(e0,u1\_0)**

**ans =**

**-375**

**-4841.2**

**4.4236e+06**

8. Describe the difference between inputs used to calculate the epipolar line in questions 1.3.5

and 1.3.7

**In question 1.3.5, a epipole and and another image plane point were used to derive the epipolar line, in 1.3.7, the Fundamental matrix and a image plane point were used to derive the epipolar line. I demonstrated the same this with my check work example in 1.3.7 using the cross product between the epipole and the other image point to derive the epipolar line instead of using the Fundamental matrix and the image point to derive the epipolar line.**

1.2.4 Reconstruction

I have implemented the equation from chapter 12.2 of Hartley and Zisserman. We did the derivation

in class of how to set up the matrix to use two corresponding image points together with the camera

matrices in order to reconstruct the 3D point. Using the image coordinates you have computed,

use the function I have written to convince yourselves that it is possible to correctly reconstruct

the 3D points if you are given the corresponding image points and the camera matrices.

**Using the 3D reconstruction, I came up with this example table (values less than 1x10^10 have been truncated to zero):**

**X0 derived from the camera matrices and u0\_0 and u0\_1**

**0**

**0**

**10**

**X1 derived from the camera matrices and u1\_0 and u1\_1**

**0**

**3**

**10**

**X2 derived from the camera matrices and u2\_0 and u2\_1**

**0**

**2.4253**

**8.0843**

**X3 derived from the camera matrices and u3\_0 and u3\_1**

**0**

**3.5747**

**11.9157**

**X4 derived from the camera matrices and u4\_0 and u4\_1**

**0.9274**

**2.4253**

**8.3238**

**X5 derived from the camera matrices and u5\_0 and u5\_1**

**-0.9274**

**3.5747**

**11.6762**

This is the OpenCV camera calibration / camera geometry documentation. Some of it is better

documented than other parts of it. http://docs.opencv.org/modules/calib3d/doc/camera\_

calibration\_and\_3d\_reconstruction.html

It is possible to recover the camera matrices, given only the 3D and 2D points. This is imple-

mented in OpenCV, but the documentation is poor. You can implement it for yourself if you would

like to see it work. There is a handout on Piazza from Chapter 7 of Hartley and Zisserman.

Finally, the Fundamental matrix can be computed from image correspondences only. This is

implemented in OpenCV, but you need at least 8 points. The points provided in this assignment are

all co-planar and will give a degenerate solution. If you would like to see the Fundamental Matrix

computation working, you should make up some extra 3D points, compute their image coordinates,

and use all the image points together.

4