Computer Vision
Exercise 2 report

Amrollah Seifoddini

STD_ID = 13-923-958

- The images are taken from Rabeeh Karimi, as I do not have a working camera yet.

Part 1)

I used the Bouget's toolbox for un-distorting the images. Firstly, I put the calibration result file from last assignment to the images folder and then run the toolbox. After loading the images, I clicked on un-distorting and the got the results. The original and un-distorted images are in the images folder attched.

Part 2)

For this part, I mainly used the exercise slides guideline to computer the F and Fh. I used the 8-point algorithm to get the F and then apply the singularity constraint to it and get the Fh. The results are as follows:

F =

-0.0143 -0.0974 -0.1637

-0.0482 0.0433 -0.5222

-0.0204 0.8283 -0.0071

Fh =

-0.0000 -0.0000 0.0000

-0.0000 0.0000 -0.0010

0.0002 0.0017 -0.3665

The overlaid images with epipolar lines for corresponding points in two images are displayed here:

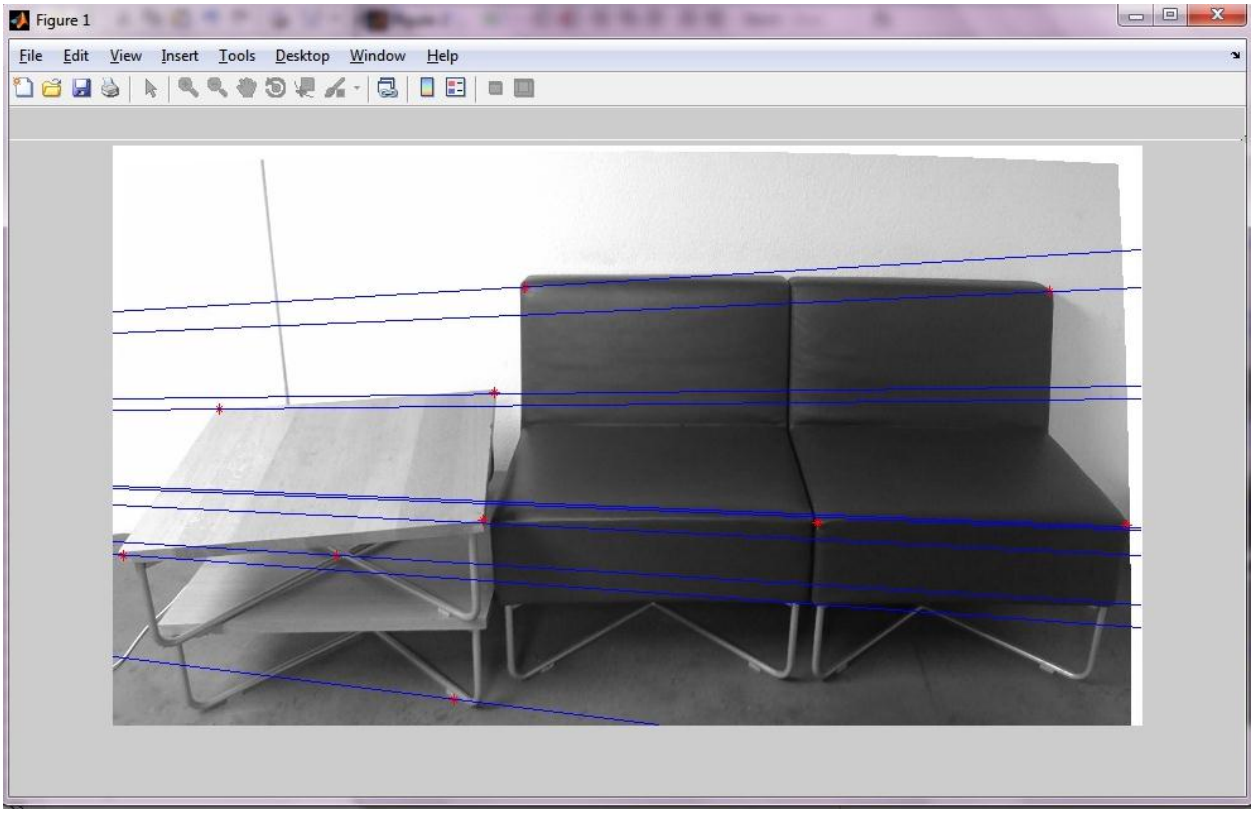
Figure 1
File Edit View Insert Tools Desktop Window Help

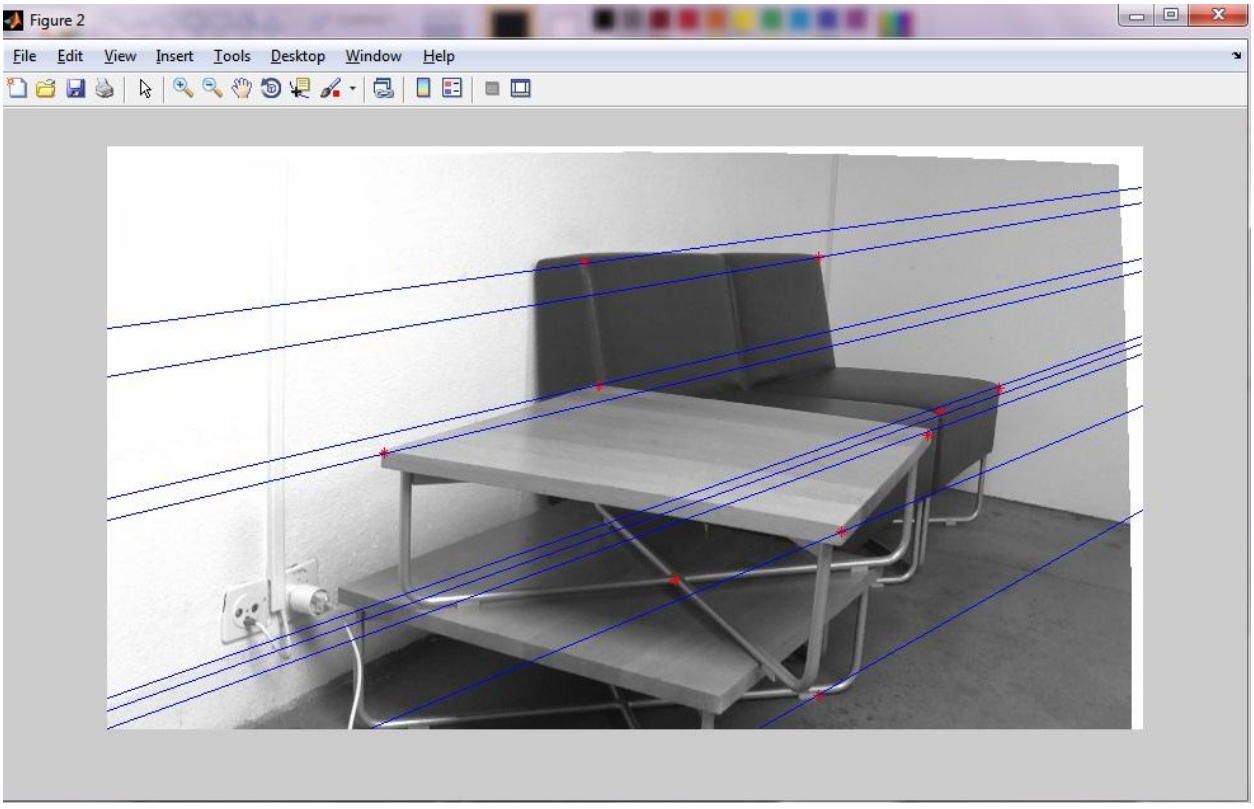
Figure 2
File Edit View Insert Tools Desktop Window Help

Part 3)

My calibration matrix from last assignment was:

```
K = 1.0e+03 * [3.5341 0 1.8105; 0 3.5633 0.6595; 0 0 0.0010];
```

For this part I also used the same algorithm as part 2. However, before passing the point, I transformed them with calibration matrix. Of course, the result is de-normalized too.

E =

0.0450	0.5552	0.0868
0.1924	-0.2651	0.5288
-0.0003	-0.5433	0.0141

Then I enforced the singularity constraint to get the Eh.

Eh =

0.0709	0.4864	0.1679
0.2306	-0.1641	0.6248
-0.0225	-0.4638	-0.0392

Again, the result images are shown here:

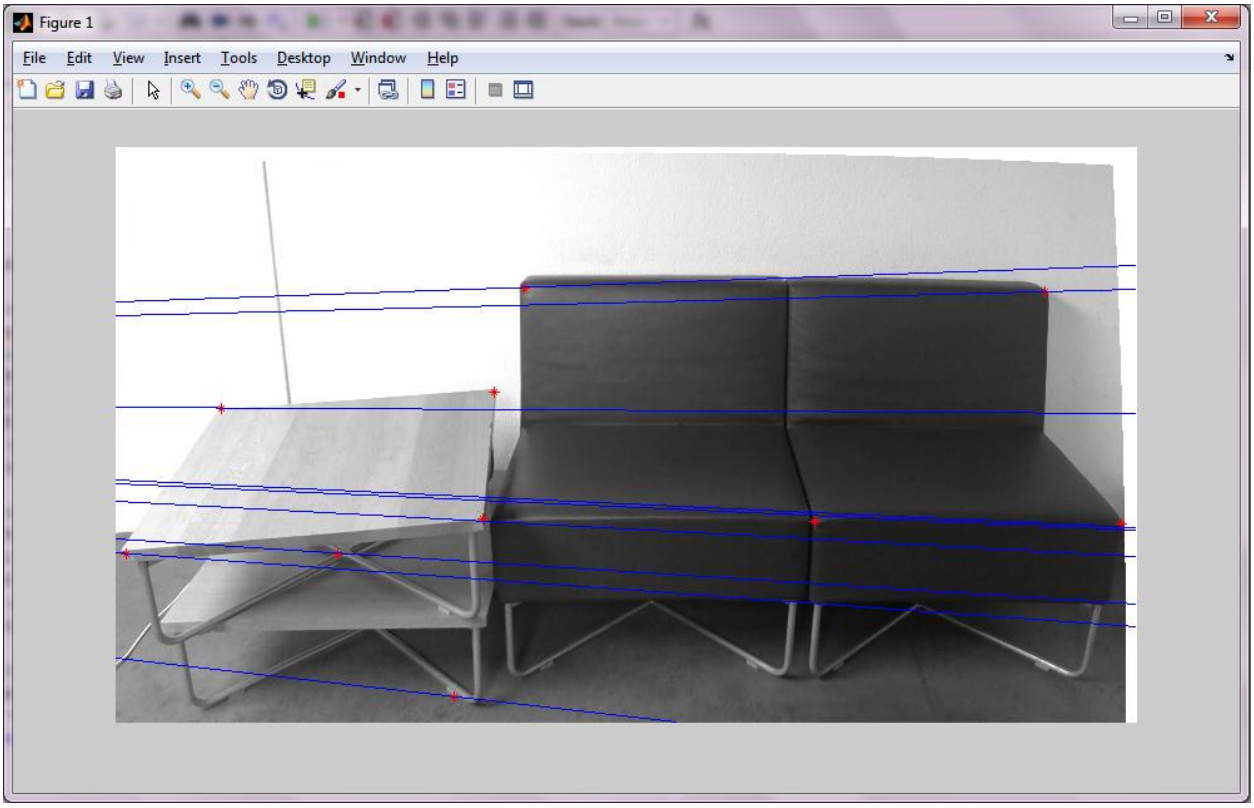
Figure 1
File Edit View Insert Tools Desktop Window Help

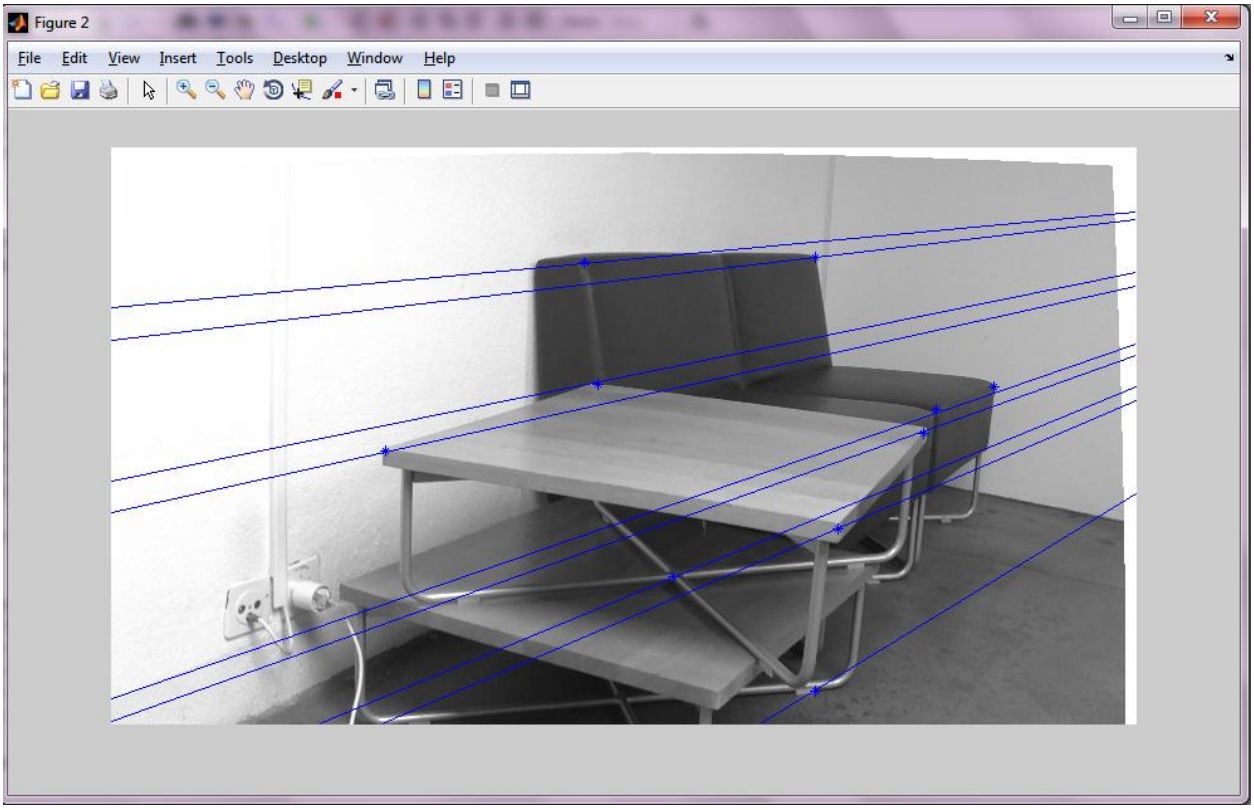
Figure 2
File Edit View Insert Tools Desktop Window Help

Part 4)

For computing the camera rotation and transition matrixes, the t is the left null vector of E and I used SVD to decompose E and get the two possible Rotation matrixes. Then I formed four solutions for camera matrix. I tested one of my points in the image with each Pcam to see if it is visible in both cameras or not. I used the triangulation function in the template source to get there-projected 3d point. Then I computed the depth of that point in each camera. If it is positive in both cameras then the current camera matrix is ok and we return it.

Pcam =

0.3767	0.2528	-0.8912	0.6606
-0.0256	0.9645	0.2627	-0.1311
0.9260	-0.0762	0.3698	0.7392

The position of two cameras is shown on a line here. I just used the showCamera function with the Pcam matrix as a cell array.