3D Computer Vision Project 2

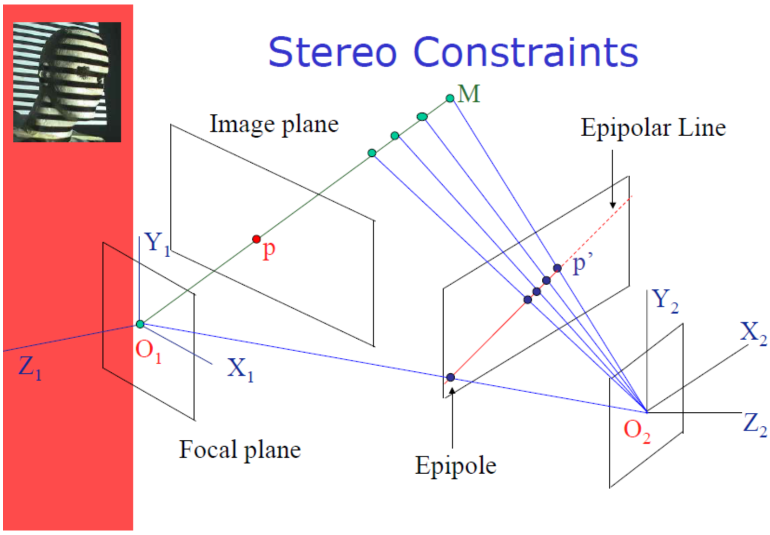
Spring semester, 2015
Bodong Zhang
u0949206

Practical Problems

4a Epipolar geometry from F-matrix

Introduction:

In this topic, we discuss the corresponding points in the left camera and right cameras if both cameras take a picture of same object. If we knew the coordinates of a point in camera 1, then all the possible locations of the point in camera 2 forms the epipolar line. The theory is shown below[1]:



In this topic, we discuss such correspondence and evaluate the results.

Algorithm:

First we take two pictures of the same object at different locations, then we designate a set of corresponding pixel landmark pairs in the two pictures for finding matrix F. Because O,O' and P is in the same plane,

$$\overrightarrow{Op} \cdot [\overrightarrow{OO'} \times \overrightarrow{O'p'}] = 0$$

We can rewrite it as $\boldsymbol{p} \cdot [\boldsymbol{t} \times (\mathcal{R}\boldsymbol{p}')] = 0$

$p = K_1^{-1}u$

Also $p' = K_2^{-1}u'$, where p is coordinates in real world, u is coordinates in image.

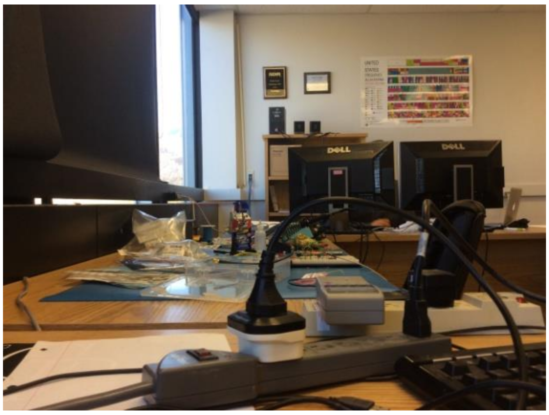
So the relationship between u and u' is like $u^T F u' = 0$, where F is fundamental matrix.

If we have known matrix F and coordinates in one camera, then (Fu') or (u^TF) is known, which could be regarded as the parameter of a straight line. Thus we can get the epipolar line.

There is a special point called epipole in camera screen because every epipolar line constraints on this point, and it is also the mapping of focal of the other camera.

Experiment:

First we took two pictures of same objects at different locations and angles.




Then for calculating matrix F, we specify a set of corresponding pixel landmark pairs as shown below. The first column is x coordinate of image1, second is y of image1, third is x of image2, fourth is coordinates of image2.

	1	2	3	4
1	1.1705e+03	814.5000	758.5000	42.5000
2	954.5000	878.5000	510.5000	134.5000
3	1.0105e+03	1.0545e+03	658.5000	378.5000
4	2.3905e+03	962.5000	2.0745e+03	222.5000
5	1.6185e+03	806.5000	1.3065e+03	58.5000
6	2.2865e+03	1.4225e+03	2.3665e+03	898.5000
7	1.0065e+03	1.2225e+03	806.5000	662.5000
8	942.5000	1.6625e+03	1.1905e+03	1.5025e+03
9	674.5000	1.7225e+03	850.5000	1.5945e+03
10	190.5000	1.6465e+03	70.5000	1.3945e+03
11	1.7945e+03	1.3745e+03	1.9425e+03	942.5000
12	1.6945e+03	1.2145e+03	1.5065e+03	582.5000
13	2.2585e+03	1.0025e+03	2.1305e+03	322.5000
14	2.3905e+03	870.5000	2.0905e+03	114.5000
15	1.9945e+03	1.5865e+03	2.1985e+03	1.1785e+03
16	654.5000	1.3905e+03	446.5000	898.5000
17	1.0465e+03	1.5705e+03	1.0665e+03	1.1745e+03

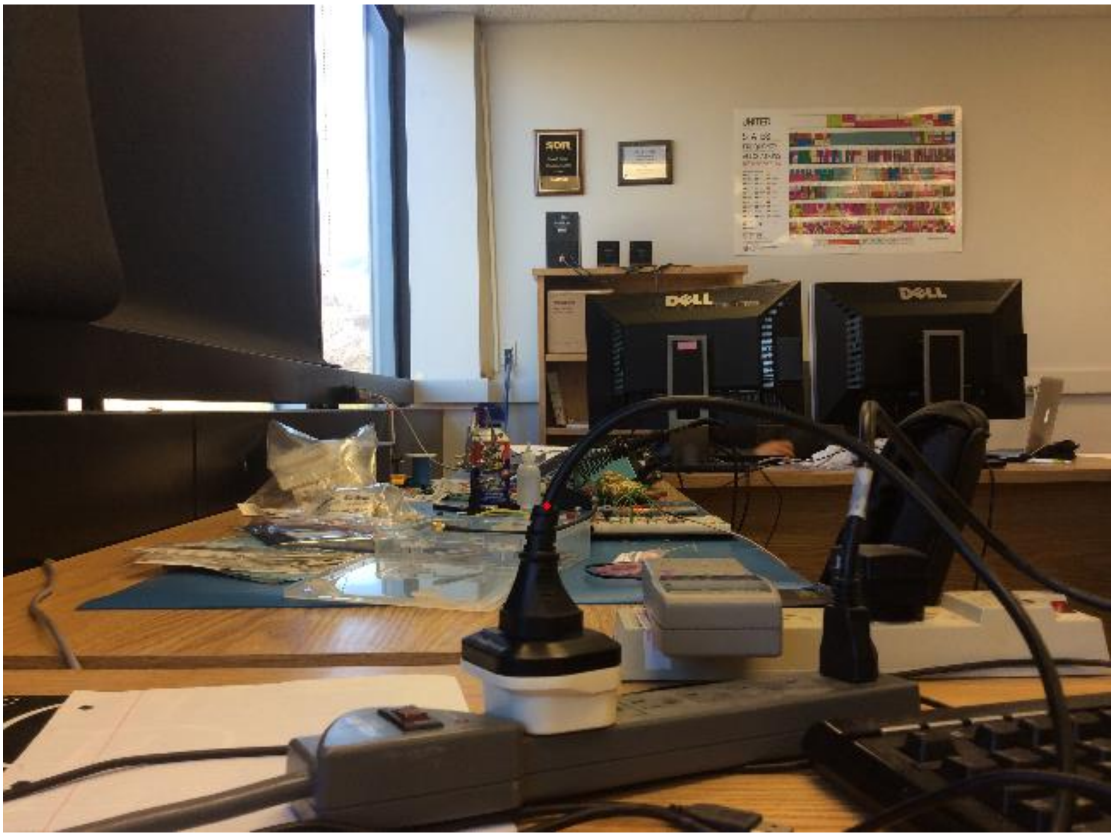
It is a least square problem to get F. The algorithm we use is that we want to minimize $\sum_{i=1}^{n}(\boldsymbol{p}_i^T \mathcal{F} \boldsymbol{p}_i')^2$

The experiment result of F is

F_matrix <9x1 double>

	1	2
1	1.4232e-08	
2	-9.5824e-07	
3	-9.9637e-04	
4	9.7549e-07	
5	-3.4587e-07	
6	0.0019	
7	1.1935e-04	
8	-7.5215e-04	
9	-1.0000	
10		

Then we could do experiment to test the experiment result. First we choose a point in image1(marked by a red point in the middle of image)



Then by multiply the matrix F with coordinates, we can get the parameter of straight line and show it in image2.

Next we choose another point in image 2 and find the epipolar line in image1. (The point is at the back of computer)

The epipolar line is shown below:

The last experiment is to find the epipole. In theory the epipole fulfill the equation of $F * e_L = 0$. By using the algorithm in the slides, we find the least eigen vector associated with least eigen value of (F'*F).

```
u =

   -0.0001    0.5542   -0.8324
    0.0008   -0.8324   -0.5542
    1.0000    0.0007    0.0003


d =

    1.0000         0         0
         0    0.0000         0
         0         0    0.0000
```

F	[1.4232e-08,-9.5824e-07,-9.9637e-04;9.754...	-1.00...	0.0019
F_matrix	[1.4232e-08;-9.5824e-07;-9.9637e-04;9.754...	-1.00...	0.0019
d	[1.0000,0,0;0,4.6657e-12,0;0,0,4.1276e-14]	0	1.0000
u	[-1.1935e-04,0.5542,-0.8324;7.5215e-04,-0....	-0.83...	1.0000

Then the epipole is below. We can see from the coordinates that it is not on the screen.

```
uu =

   1.0e+003 *

   -2.6213
   -1.7455
    0.0010
```

Evaluation:

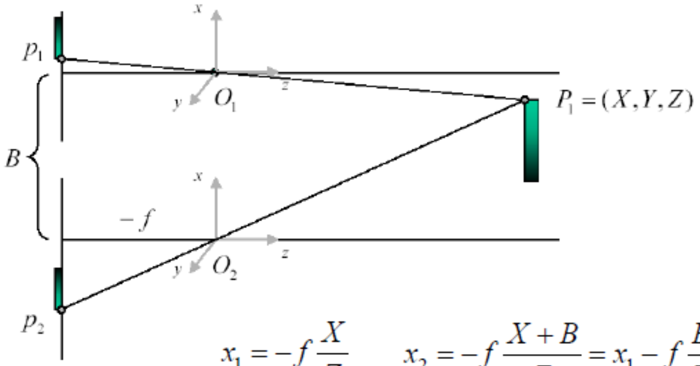
From the results of epipolar line, we can see that it makes sense because the corresponding point locates at the epipolar line.
Also, we draw two epipolar lines in the same image. By estimation, the location of epipole conforms with calculation result.

Problem 4b Dense distance map from dense disparity

Algorithm:

In this problem, we take two pictures by horizontal movement of a camera. According to the geometry theory, the corresponding points in two images have the same vertical coordinate. So while finding corresponding point, we only need to search in one line.

The theory of calculating depth is[1]



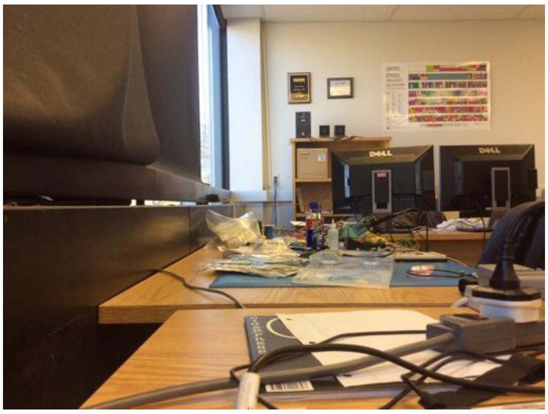
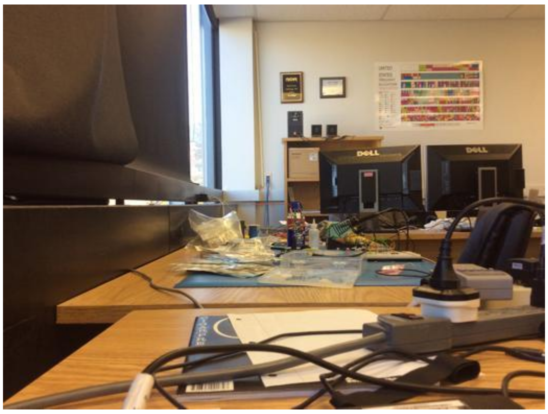
$$x_1 = -f\frac{X}{Z} \qquad x_2 = -f\frac{X+B}{Z} = x_1 - f\frac{B}{Z}$$

$$Z = \frac{fB}{x_1 - x_2}$$

Z is called depth. We also need the focal length and movement distance B and pixel size in camera to get depth besides the coordinates of point in two cameras.

Experiment:

First we take two pictures described above.

The horizontal distance of cameras is 65mm. Then for each location of left image, we find the best matched location in right image by calculating the normalized cross-correlation of window:

Normalized cross-correlation:

$$C(d) = \frac{1}{\|\boldsymbol{w} - \bar{\boldsymbol{w}}\|} \frac{1}{\|\boldsymbol{w}' - \bar{\boldsymbol{w}}'\|} [(\boldsymbol{w} - \bar{\boldsymbol{w}}) \cdot (\boldsymbol{w}' - \bar{\boldsymbol{w}}')].$$

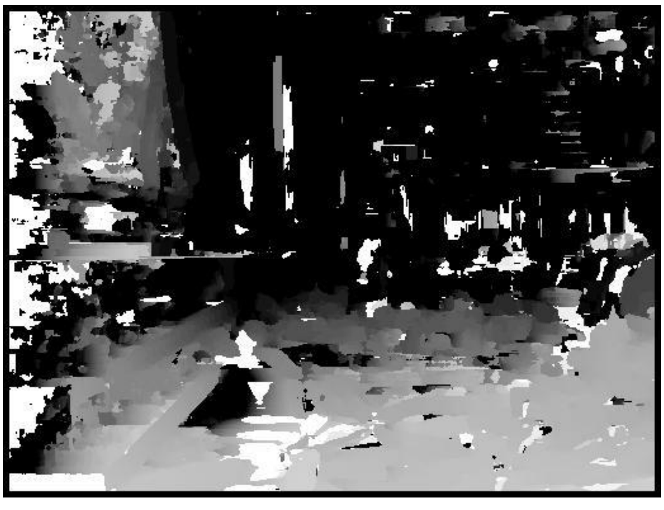
The advantage of this method is that it is invariant to intensity change like I'=aI+b, so the influence of light is low.

While searching for the correlation, we could use different size of window. In this experiment, 3*3, 5*5, 11*11 window sizes are used to compare results. After getting the disparity(the difference of two x coordinates), we could deduce the depth of each pixel.

The experiment results are below. Bright color means that the depth is short, dark color means that the depth is long.

Depth(window size 3*3)

Depth(window size 5*5)



Depth(window size 11*11)

We can see that if the searching window is large, it is less influenced by noise and makes the results stable.

According to the data, the depth is usually among 1 meter to 10 meters, which makes sense to the situation.

References

[1]slides in class