cs512 Assignment 4

Jiranun Jiratrakanvong
Department of Computer Science
Illinois Institute of Technology

Dec 1, 2015

Abstract

The objective of this assignment is to learn about stereo vision algorithm by conducting a programs using python as a programming language. The programs of this assignment will locate and drawing epipolar lines and epipoles of an uncalibrated stereo pair.

Problem Statement

The program from this assignment need to be able to calculate fundamental matrix from at least 8 initial corresponding points. Then, the epipoles of left and right images, and epipolar lines corresponding to selected points will be drawn according to the fundamental matrix.

Purpose Solution

Users can specify pairs of corresponding points through input file, or specify by choosing on the left and right images. For manual choosing, the program has mouse callback for both left and right images. When a location of an image is clicked, the program will draw a point on that location along with its number. Number of corresponding points can be specified through a parameter **INIT_POINTS** under section **ASG4** in **Asg4.config**.

```
if event == cv2.EVENT_LBUTTONDOWN:
    if not alreay_init:
        if L_point_left > 0:
            L_init_points.append((float(x),float(y)))
            cv2.ellipse(img_L, (x,y), (1,1), 0, 0, 360, 0, 2)
            cv2.putText(img_L, str(nInitPoints+1-L_point_left),
                        (x,y-4), cv2.FONT_HERSHEY_PLAIN, 1.0, (0,0,0), 1)
            L_point_left -= 1

        if L_point_left == 0 and R_point_left == 0:
            alreay_init = True
```

Part of mouse callback for the left image

When all corresponding points have been specified, the program will calculate the fundamental matrix in order to display epipoles and epipolar lines. The steps of finding fundamental matrix include

1. **Normalize point sets** : subtract each corresponding point by the mean, and divide by the variance. Then, calculate normalizing matrix for the last step.

```
# normalize points
mean_L = (np.average([x for (x,y) in L_init_points]), np.average([y for (x,y) in L_init_points]))
var_L = (var([x for (x,y) in L_init_points]), var([y for (x,y) in L_init_points]))
norm_left_points = np.array([ ((p[0] - mean_L[0])/var_L[0],(p[1] - mean_L[1])/var_L[1]) for p in L_init_points], np.float32)
v_mat = np.array([[1./var_L[0],0.,0.],[0.,1./var_L[1],0.],[0.,0.,1.]])
m_mat = np.array([[1.,0.,-mean_L[0]],[0.,1.,-mean_L[1]],[0.,0.,1.]])
M_L = np.dot(v_mat, m_mat)

mean_R = (np.average([x for (x,y) in R_init_points]), np.average([y for (x,y) in R_init_points]))
var_R = (var([x for (x,y) in R_init_points]), var([y for (x,y) in R_init_points]))
norm_right_points = np.array([ ((p[0] - mean_R[0])/var_R[0],(p[1] - mean_R[1])/var_R[1]) for p in R_init_points], np.float32)
v_mat = np.array([[1./var_R[0],0.,0.],[0.,1./var_R[1],0.],[0.,0.,1.]])
m_mat = np.array([[1.,0.,-mean_R[0]],[0.,1.,-mean_R[1]],[0.,0.,1.]])
M_R = np.dot(v_mat, m_mat)
```

2. **Do 8 points Algorithm on the normalized corresponding points** : Create matrix A from normalized corresponding points. Then, get the last column of V, or last row of V^T which is from SVD of A

A = $$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & y_m & x'_m & y'_m & 1 \end{bmatrix}$$

```
# build Fundamental matrix (F') by init points by normalized points
A = np.array([[xl*xr, xl*yr, xl, yl*xr, yl*yr, yl, xr, yr, 1.]
              for ((xl,yl),(xr,yr)) in zip(norm_left_points, norm_right_points)])
U, D, VT = np.linalg.svd(A)
F_p = VT[-1,:]
F_p = np.array([[F_p[0],F_p[1],F_p[2]],[F_p[3],F_p[4],F_p[5]],[F_p[6],F_p[7],F_p[8]]])
```

3. **Enforce rank 2 constraint F** : find SVD of F` then replace the smallest singular value in D by 0, and calculate the new F` by F` = UD`V^T

```
# Enforcing rank 2 contraint F'
U, D, VT = np.linalg.svd(F_p)
D[np.argmin(D)] = 0
F_p = np.dot(U,np.dot(np.diag(D),VT))
```

4. **Compute the fundamental matrix** : using normalizing matrix from step 1. to find the final F.

$$F = M_L^T \cdot F` \cdot M_R$$

```
# Compute F from F'
F = np.dot(M_L.T, np.dot(F_p, M_R))
```

After the fundamental matrix (F) calculated, the program will find the epipoles, and display F and epipoles to console. Then, the epipoles will be draw on the left and right images.

```
#Find Epipoles
U, D, VT = np.linalg.svd(F.T)
last_col_u = U[:,-1]
e_r = (int(np.round(last_col_u[0]/last_col_u[2])),int(np.round(last_col_u[1]/last_col_u[2])))
last_col_v = VT[-1,:]
e_l = (int(np.round(last_col_v[0]/last_col_v[2])),int(np.round(last_col_v[1]/last_col_v[2])))
```

Finding epipoles

After epipoles found and displayed, the mouse callback of each image will be called again when user click on any location of any image. At this time, the mouse callback will draw a point on the location clicked, and draw the epipolar line corresponding to that location on another image. The line parameter will be calculated by the fundamental matrix and the coordinates. The formula will be $F \cdot P$ or $F^T \cdot P$ depends on the side of the line.

```
def draw_line(p, side):
    global F, img_L, img_R
    rand_color = (r(),r(),r())

    if side == 'right':
        line = np.dot(F.T,p)
        [a,b,c] = line
        line_angle = np.arctan(b/a)

        if np.abs(line_angle) < 1.:
            for y in range(img_R.shape[0]):
                x = int(np.round((-c-b*y)/a))
                if x in range(img_R.shape[1]):
                    cv2.ellipse(img_R, (x,y), (0,0), 0, 0, 360, rand_color, 2)
        else:
            for x in range(img_R.shape[1]):
                y = int(np.round((-c-a*x)/b))
                if y in range(img_R.shape[0]):
                    cv2.ellipse(img_R, (x,y), (0,0), 0, 0, 360, rand_color, 2)

        cv2.ellipse(img_L, (int(p[0]),int(p[1])), (1,1), 0, 0, 360, rand_color, 2)
```

draw_line function

Implementation Details

Problem and Solution

- There are many steps to calculate the fundamental matrix, so it was hard to check if the fundamental matrix was correct. The solution is that making sure that the PFP` = 0, and also trying to draw the epipolar line by calculated F.
- The epipoles are not at the intersection of epipolar lines. The solution is that using the transpose of F.

Instructions for using the program

- Config number of corresponding points for 8 points algorithm via parameter **INIT_POINTS** under **[ASG4]** in **Asg4.config**

- Running via command line :

python CS512Asg4.py [left image] [right image]

or

python CS512Asg4.py [left image] [right image] [corresponding points (optional)]

  - If the corresponding points have not been specify, the program will display number of corresponding points that user have to specify.
  - Each line of the corresponding points file will be x_l y_l x_r y_r separated by space. For example,

```
39.0 20.0 34.0 18.0
84.0 60.0 81.0 59.0
107.0 81.0 105.0 81.0
122.0 95.0 121.0 93.0
161.0 94.0 159.0 92.0
176.0 68.0 172.0 66.0
206.0 13.0 200.0 13.0
198.0 53.0 193.0 52.0
196.0 153.0 192.0 154.0
203.0 215.0 198.0 215.0
179.0 174.0 177.0 174.0
```

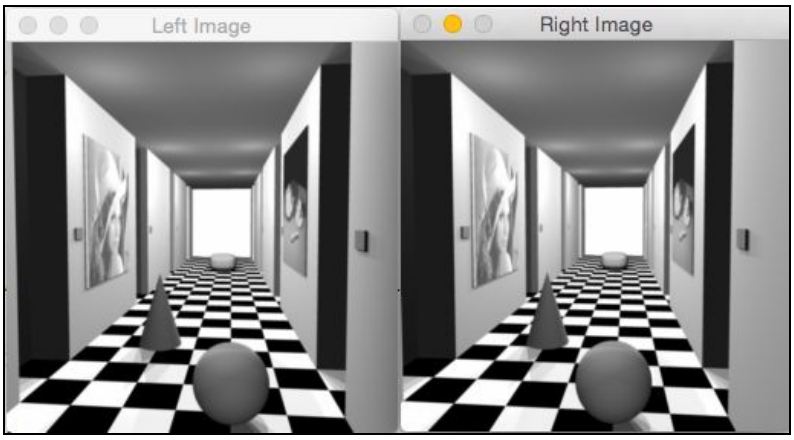
    If number of corresponding points are less than 8, users need to specify them manually.

- During specify the corresponding points, the program will show that points marked with its number.
  - Press ‘h’ for help
  - Press ‘r’ to restart setting corresponding points
  - Press <ESC> to quit the program

- After corresponding points specified
  - Press ‘h’ for help
  - Press ‘r’ to restart setting corresponding points
  - Press ‘c’ to clear the images
  - Press ‘e’ to display the epipoles
  - Press <ESC> to quit the program

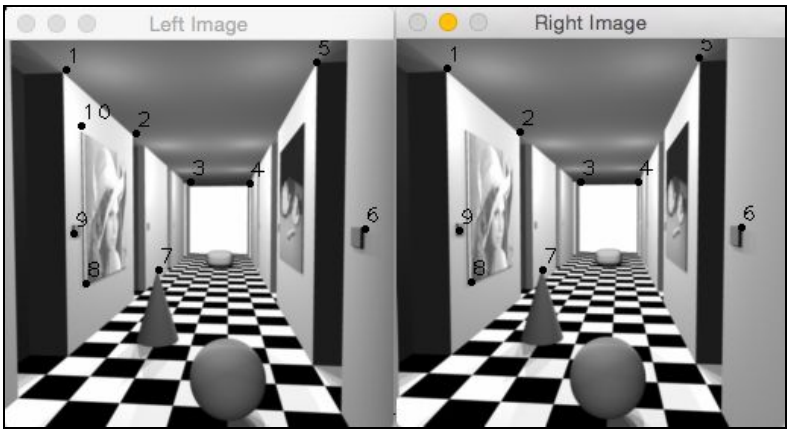
Results and discussion

After the program started, if the file of corresponding points was not specified. The program will display left and right images, and show number of points that user has to specify on images manually.



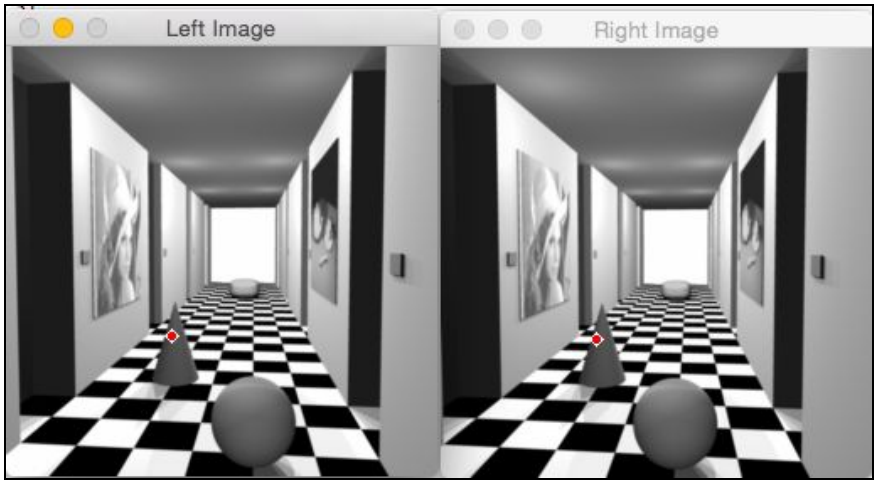

```
Please click 10 corresponding points on left and right images.
```

When user click any image, the program will draw a point and its number respectively. User has to provide all corresponding points before the result shown.

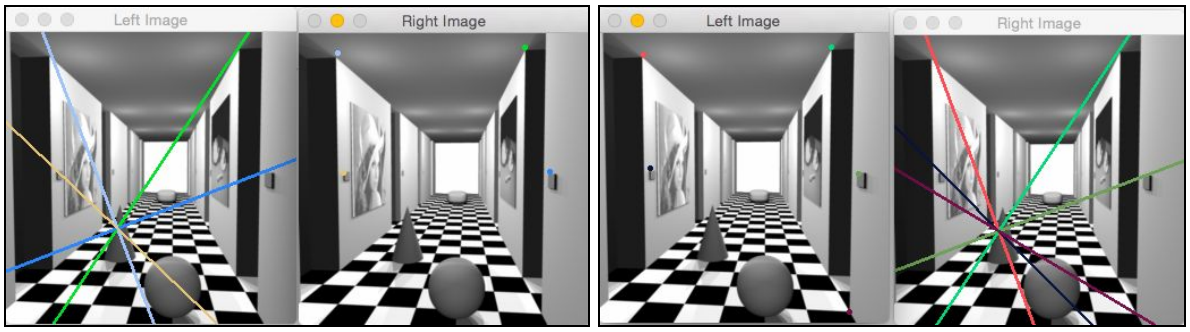


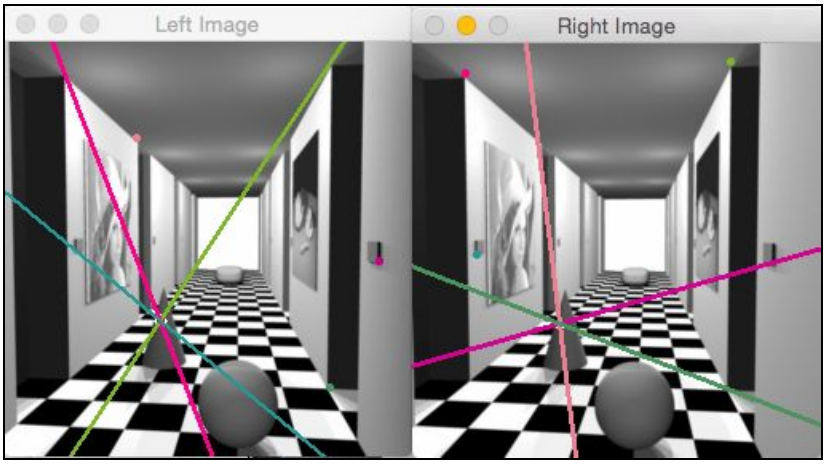
After all corresponding points have been specified, the program will print the estimated fundamental matrix and the coordinates of epipoles. Also, the location of epipoles will be shown on both images with red dots.

```
Estimated Fundamental Matrix :
[[  4.24839086e-10   3.62980181e-08  -6.30646471e-06]
 [ -3.67824126e-08  -2.74888450e-10   3.43063105e-06]
 [  6.27726524e-06  -3.50166184e-06   2.72442267e-05]]
The coordinates of the left epipole : (92, 173)
The coordinates of the right epipole : (98, 172)
```



When user click a point of any image, the corresponding epipolar line will be drawn on another image by using the same color as the point.



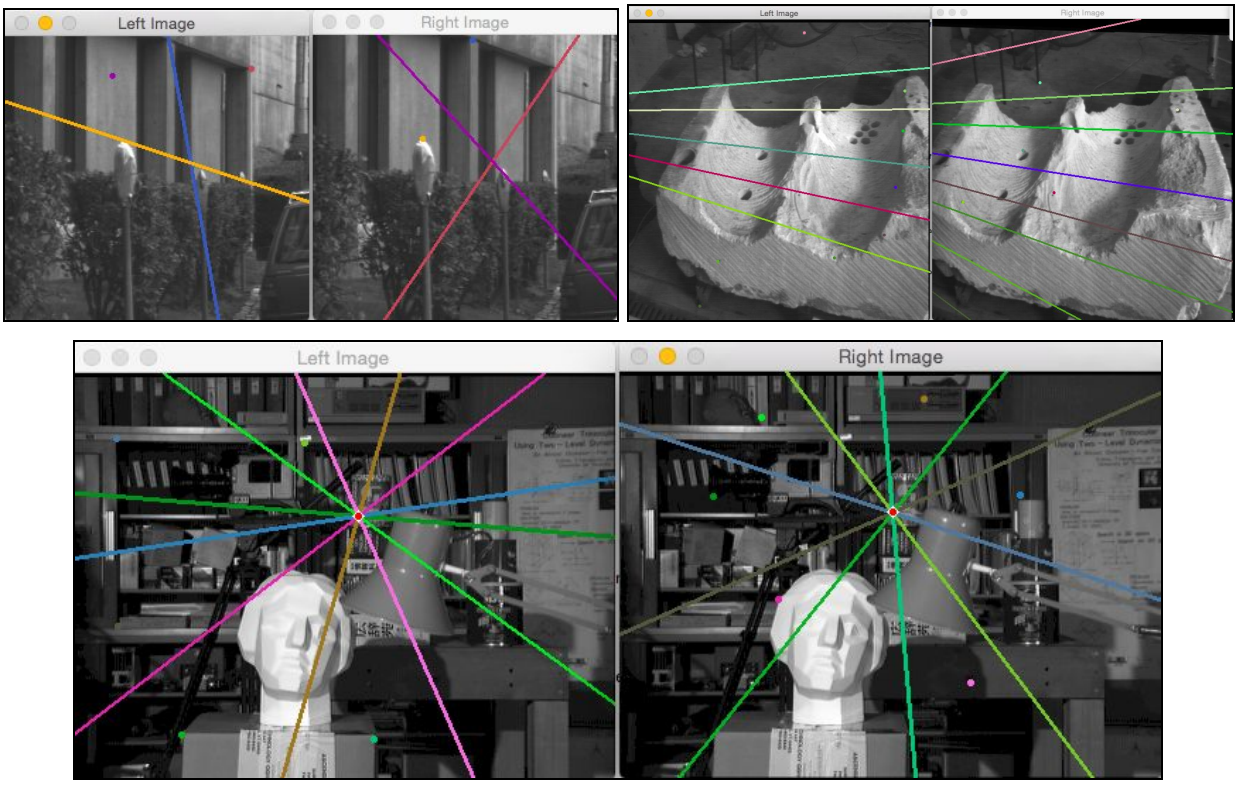


Strength and weaknesses

The corresponding points can be both specified through a file or manually, so user does not need to specify them every time the program run. Moreover, the color of epipolar line and its point will be the same, so it's easy to see. However, the weakness is that if number of the corresponding points are not enough the line will not be accurate.

Evaluation

Testing by many stereo vision images from http://www.ijrr.org/historic/contents/20_07/abstract/512.html



References

- Gady Agam - *CS512: Computer Vision, Introduction to Python, and Using Python with OpenCV*, Department of Computer Science, Illinois Institute of Technology
- Robert Collins - *The Eight-Point Algorithm* http://www.cse.psu.edu/~rtc12/CSE486/lecture20_6pp.pdf
- Stereo pairs http://www.ijrr.org/historic/contents/20_07/abstract/512.html