CS512 Assignment 5: Epipolar Geometry

Adithya Sreenath Chandrashekarapuram

Department of Computer Science

Illinois Institute of Technology

November 26, 2017

Abstract:

Epipolar geometry is the geometry of stereo vision. When two cameras view a 3D scene from two distinct positions, there are a number of geometric relations between the 3D points and their projections onto the 2D images that lead to constraints between the image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model.

Implementation:

- We first take a set of stereo images as the input.

```
Img_var = cv2.imread (image_name)
```

- Then we allow the user to select 8 points on each image.
- We find the fundamental matrix by using these 8 points on each image.

1. We pass the points which are stored in an array to fundamental matrix function

```
def fundamental_matrix(*args):
  try:
    x1,x2,npts = check_input(args)
    F = eight_point_algorithm(x1,x2)
    return F
  except ErrorShape, e:
    print 'ErrorShape, exception message:', e
    return None
```

2. This function checks for validity of input and gives out an error message if inputs are wrong.
3. If the correct inputs are provided then the 8 point algorithm is executed

```
def eight_point_algorithm(x1,x2):

  # perform the normalization
  x1, T1 = normalize2dpts(x1);
  x2, T2 = normalize2dpts(x2);
  A = constraint_matrix(x1,x2)
  (U, S, V) = linalg.svd(A)
  V = V.conj().T;
```

```
F = V[:,8].reshape(3,3).copy()

#F should be of rank 2
(U,D,V) = linalg.svd(F);
F = dot(dot(U,diag([D[0], D[1], 0])),V);

#Denormalize
F = dot(dot(T2.T,F),T1);
return F
```

- Inside the 8 point algorithm we need to normalize the matrices

```
def normalize2dpts(pts):
    if pts.shape[0]!=3:
        raise ErrorShape('points must be 3xN')

    finiteind = abs(pts[2]) > finfo(float).eps
    pts[0,finiteind] = pts[0,finiteind]/pts[2,finiteind]
    pts[1,finiteind] = pts[1,finiteind]/pts[2,finiteind]
    pts[2,finiteind] = 1

    # Centroid of finite points
    c = [mean(pts[0,finiteind]), mean(pts[1,finiteind])]

    # Shift origin to centroid.
    newp0 = pts[0,finiteind]-c[0]
    newp1 = pts[1,finiteind]-c[1]

    meandist = mean(sqrt(newp0**2 + newp1**2));

    scale = sqrt(2)/meandist;
    T = eye(3)
    T[0][0] = scale
    T[1][1] = scale
    T[0][2] = -scale*c[0]
    T[1][2] = -scale*c[1]
    newpts = dot(T, pts)

    return newpts, T
```

- We then find the corresponding epipolar lines by using the fundamental matrix.

```
cv2.computeCorrespondEpilines (pts2.reshape(-1,1,2), 2,X)
```

- We display the epipolar lines on the image and give the user an option to select any of the points and display the corresponding epipolar line on the other image.

Manual:

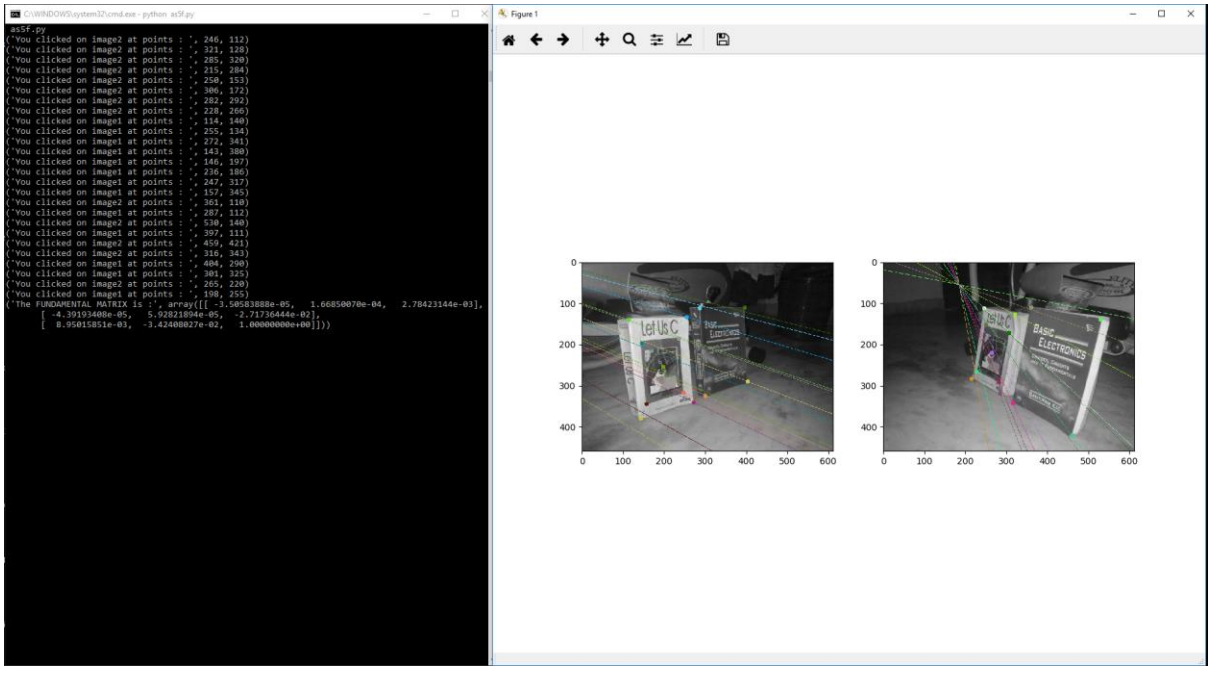
- First run the file
- Select 8 points on the image1 and select the corresponding points on image2.
- Click on ‘q’ to load the corresponding points and find the fundamental matrix.
- The epipolar lines for all the points are displayed on a plot.
- Close the plot
- 2 windows will open where the user can select points on any of the image points to load the corresponding epipolar lines in the other image.
  Note: The point selection must be accurate for the epipolar line to be displayed. Clicking multiple times inside the coloured tuple until the result is displayed is recommended.
- Click on the displayed result and press ‘q’ to close that window.
- Press ‘q’ again to close the windows and exit the program.

Results and Discussion:

2 images being displayed for selection of points:

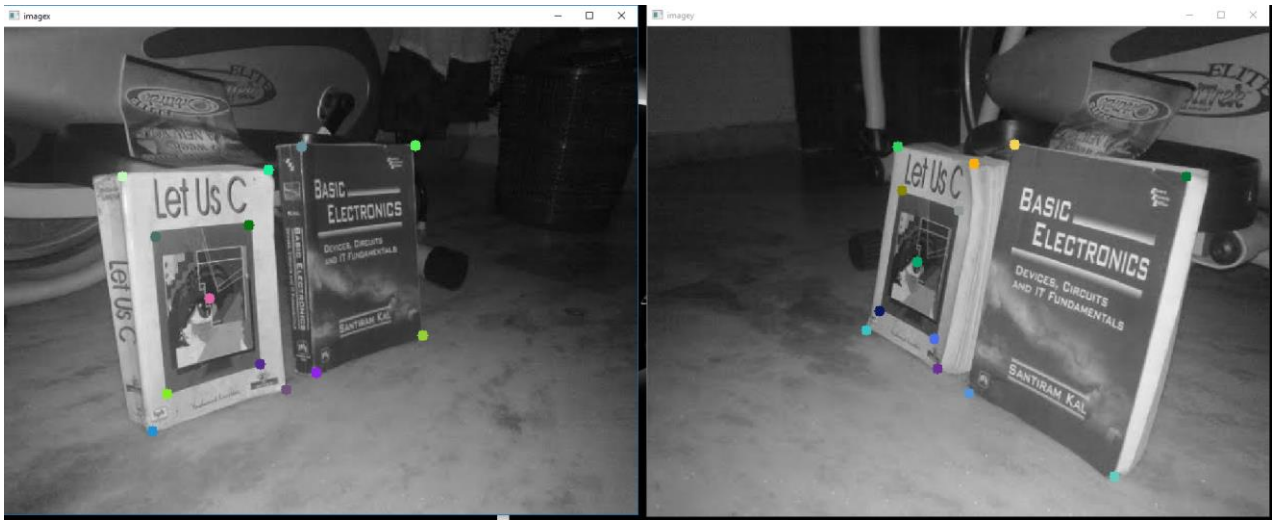
Select the desired amount of points:

The Fundamental matrix and the epilines of all the selected points:

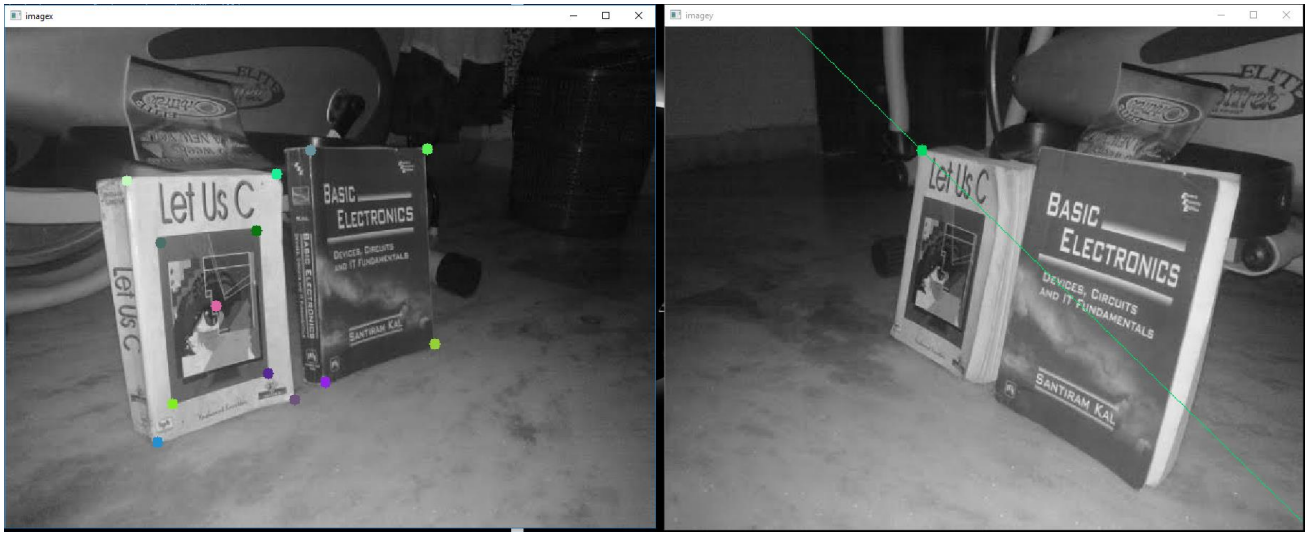


Images being displayed for selection of points:

Here the coloured dots act as reference so that the user can select the points easily.



Selection of the top left dot on the left image results in the following epiline in the right image:



The epilines coordinates:

```
('The coordinates of epipoles 2 are:', [[246, 112], [321, 128], [285, 320], [215, 284], [250, 153], [306, 172], [282, 292], [228, 266], [361, 110], [530, 140], [459, 421], [316, 343], [265, 220]])
('The coordinates of epipoles of 1 are:', [[114, 140], [255, 134], [272, 341], [143, 380], [146, 197], [236, 186], [247, 317], [157, 345], [287, 112], [397, 111], [404, 290], [301, 325], [198, 255]])
('The coordinates of epipoles 2 are:', [[246, 112], [321, 128], [285, 320], [215, 284], [250, 153], [306, 172], [282, 292], [228, 266], [361, 110], [530, 140], [459, 421], [316, 343], [265, 220]])
```

References:

https://docs.opencv.org/3.1.0/da/de9/tutorial_py_epipolar_geometry.html

https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

https://en.wikipedia.org/wiki/Epipolar_geometry