hw3

November 23, 2019

1 CSE 152 Homework 3

2 Problem 1: Perspective Projection [20 pts]

Consider a perspective projection where a point

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

is projected onto an image plane Π' represented by $k = f' > 0$ as shown in the following figure.

The first, second, and third coordinate axes are denoted by i, j, k respectively.

Consider the projection of two rays in the world coordinate system

$$Q_1 = [7\ \text{-3}\ 1] + t[8\ 2\ 4]$$

$$Q_2 = [3\ \text{-5}\ 9] + t[8\ 2\ 4]$$

where $-\infty \le t \le -1$.

Calculate the coordinates of the endpoints of the projection of the rays onto the image plane. Identify the vanishing point based on the coordinates.

Your calculation here

(For this problem, since Q_2 has a positive k, you only need to identify the vanishing points for full points)

Using pinhole camera model,

$$x' = \frac{fx}{z}$$

$$y' = \frac{fy}{z}$$

So $Q_1 = [8t+7, 2t-3, 4t+1]$ is projected to $[f'\frac{8t+7}{4t+1}, f'\frac{2t-3}{4t+1}]$; $Q_2 = [8t+3, 2t-5, 4t+9]$ is projected to $[f'\frac{8t+3}{4t+9}, f'\frac{2t-5}{4t+9}]$.

When $t \to \infty$, Q_1 is projected to $[2f, f/2]$ and Q_2 is projected to $[2f, f/2]$. So the vanishing point is $[2f, f/2]$.

(The above is all you need to show. Correctly identifying the projected point at $t = -1$ will give partial credit)

3 Problem 2: Epipolar Geometry

- (a) Suppose two cameras fixate on a point P (see the figure below) in space such that their principal axes (the line passing the optical center and along the viewing direction) intersect at that point. Show that if the image coordinates are normalized so that the coordinate origin $(0, 0)$ concides with the principal point (the intersection between the principal axes and the image plane), then the $\mathbf{F}_{33}$ element of the fundamental matrix is zero. [20 pts]

Your proof here.

Let the center of the left image plane be $p = [0, 0, 1]^T$, the center of the right image plane be $p' = [0, 0, 1]^T$. We know.

$$p^T F p' = 0$$

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{13} \\ F_{23} \\ F_{33} \end{bmatrix} = 0$$

$$F_{33} = 0$$

- Consider the case of two cameras viewing an object such that the second camera differs from the first one by a pure translation parallel to the image plane. Show that the epipolar lines in the two cameras are parallel. (hint: show by solid geometry) [10 extra credits]

Your proof here

Epipolar lines on the image planes are the intersection between epipolar plane with each of the image planes. Since image planes are parallel, epipolar plane intersect with them, then the intersection lines are parallel, so the epipolar lines are parallel.

(This is all you need to show. More explanations see below)

Let A, B be two planes such that $A//B$ (image planes). Let C be a plane intersecting A at l_1 and intersecting B at l_2 (epipolar plane). Clearly, l_1 and l_2 are co-planar. Since $A//B$, l_1 and l_2 do not intersect. Therefore $l_1//l_2$ by definition.

4 Problem 3: Fundamental Matrix [60 pts]

In this problem we will play around with sparse stereo matching methods. You will work a warrior figure. The problem contains two images ('warrior0.png', 'warrior1.png'), and two sets of matched points which is manually selected (wcor1.npy and wcor2.npy).

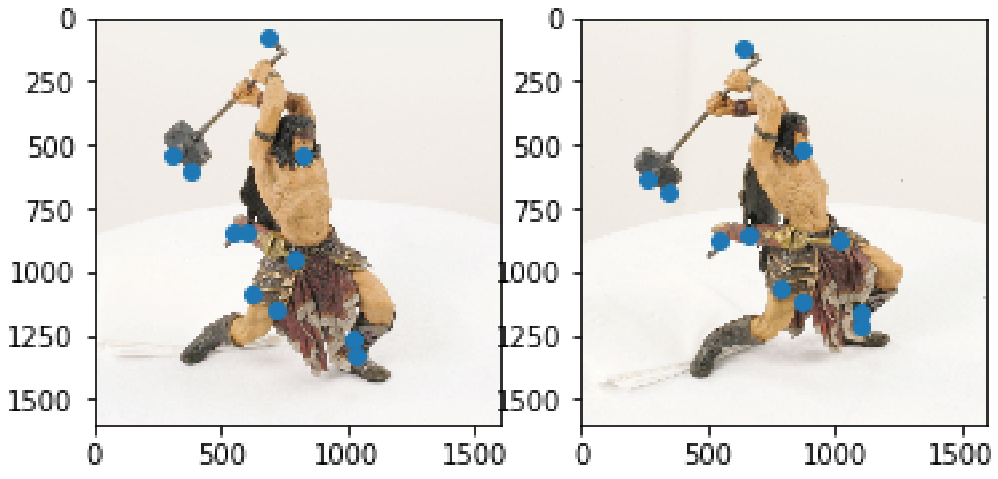
4.1 3.1 Set up

Let's first plot the images and points.

[5]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage.filters import gaussian_filter, convolve
import scipy
import warnings
from skimage.io import *
warnings.filterwarnings('ignore')
```

[6]:
```python
warriors = [imread('warrior0.png')/ 255., imread('warrior1.png') / 255.]
cor1 = np.load("wcor1.npy")
cor2 = np.load("wcor2.npy")

# plot the warriors and selected matching points
plt.subplot(121)
plt.imshow(warriors[0])
plt.scatter(cor1[0], cor1[1])
plt.subplot(122)
plt.imshow(warriors[1])
plt.scatter(cor2[0], cor2[1])
```

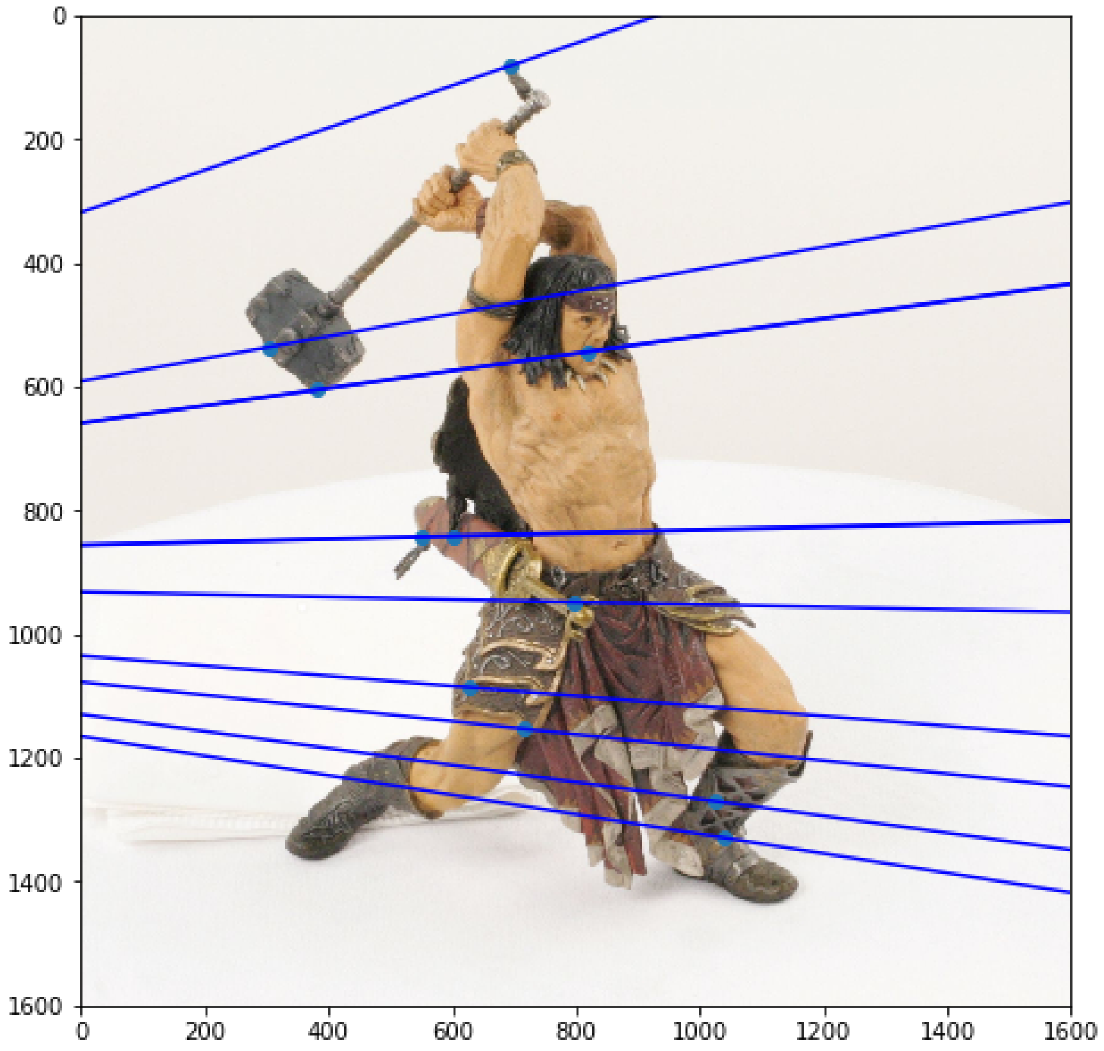
[6]:
```
<matplotlib.collections.PathCollection at 0x7f1433e48ba8>
```

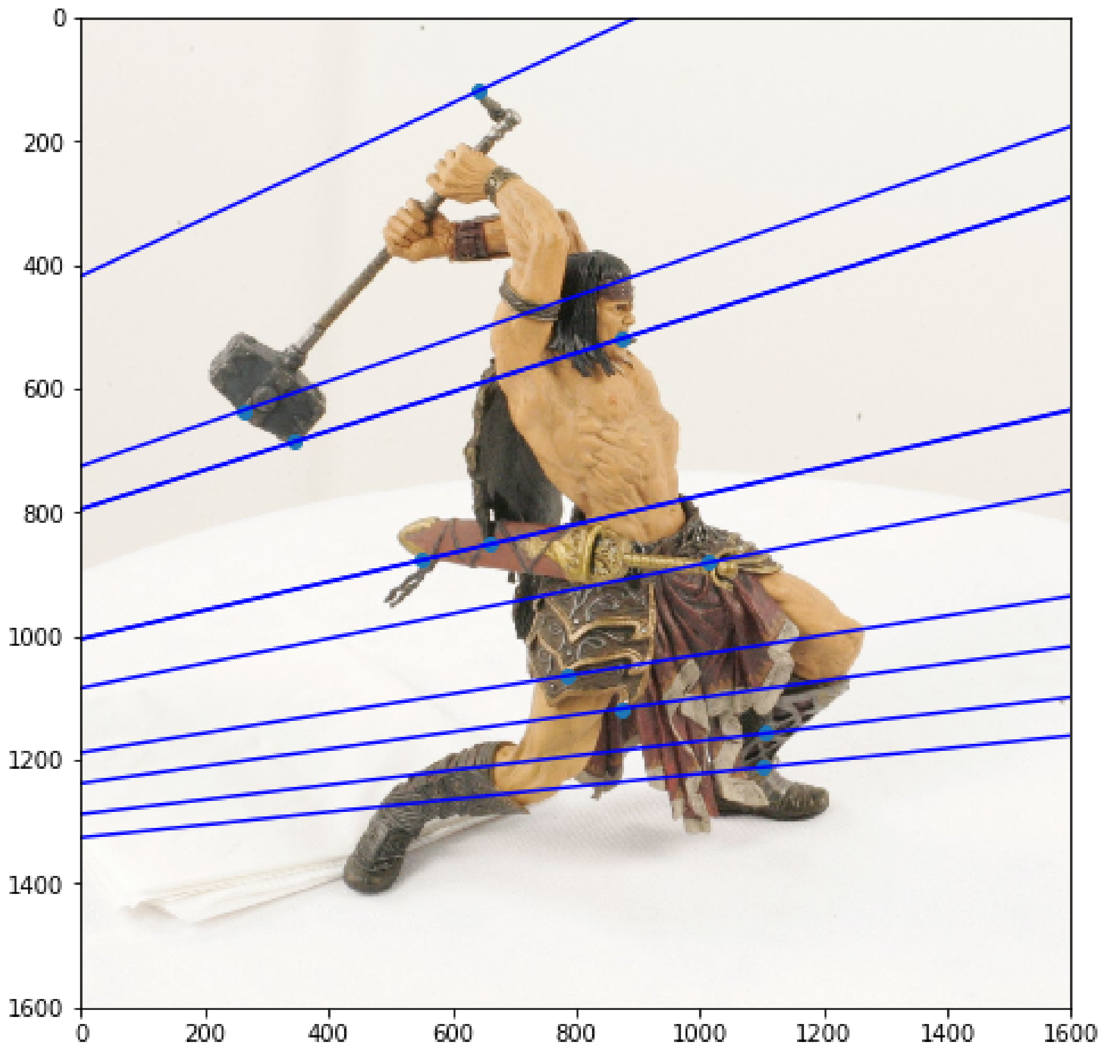


4.2 3.2 Compute fundamental matrix using 8-point algorithm [60 pts]

In this question, you need to implement 3 methods: `eight_point`, `fundamental_matrix`, `compute_epipole`. 1. In `eight_point`, you should use 8-point algorithm to estimate the fundamental matrix. However, the provided points are larger than 8, so it becomes an over-determinated linear system. So you should get a least square solution for the fundamental matrix F, and ensure $rank(F) = 2$ by set the last singular value to be zero. 2. In `fundamental_matrix`, you should perform normalization on image coordinates, and compute fundamental matrix using `eight_point`, and then perform reverse normalization. 3. In `compute_epipole`, you need to compute the epipole of two images based on the fundamental matrix 4. In `plot_epipolar_lines`, you need to plot the epipolar lines based on the above methods. You can find the reference images `epi_1.png`, `epi_2.png` to compare with your results.

[10]:
```python
import numpy as np
import matplotlib.pyplot as plt

def eight_point(x1,x2):
    """ Computes the fundamental matrix from corresponding points
        (x1,x2 3*n arrays) using the 8 point algorithm.
        Each row in the A matrix below is constructed as
        [x'*x, x'*y, x', y'*x, y'*y, y', x, y, 1]
    """

    n = x1.shape[1]
    if x2.shape[1] != n:
        raise ValueError("Number of points don't match.")

    # build matrix for equations
    A = np.zeros((n,9))
    for i in range(n):
        A[i] = [x1[0,i]*x2[0,i], x1[0,i]*x2[1,i], x1[0,i]*x2[2,i],
                x1[1,i]*x2[0,i], x1[1,i]*x2[1,i], x1[1,i]*x2[2,i],
                x1[2,i]*x2[0,i], x1[2,i]*x2[1,i], x1[2,i]*x2[2,i] ]

    # compute linear least square solution
    U,S,V = np.linalg.svd(A)
    F = V[-1].reshape(3,3)

    # constrain F
    # make rank 2 by zeroing out last singular value
    U,S,V = np.linalg.svd(F)
    S[2] = 0
    F = np.dot(U,np.dot(np.diag(S),V))

    return F/F[2,2]


def fundamental_matrix(x1,x2):
    n = x1.shape[1]
    if x2.shape[1] != n:
        raise ValueError("Number of points don't match.")
    # normalize image coordinates
    x1 = x1 / x1[2]
    mean_1 = np.mean(x1[:2],axis=1)
    S1 = np.sqrt(2) / np.std(x1[:2])
    T1 = np.array([[S1,0,-S1*mean_1[0]],[0,S1,-S1*mean_1[1]],[0,0,1]])
    x1 = np.dot(T1,x1)

    x2 = x2 / x2[2]
    mean_2 = np.mean(x2[:2],axis=1)
    S2 = np.sqrt(2) / np.std(x2[:2])
    T2 = np.array([[S2,0,-S2*mean_2[0]],[0,S2,-S2*mean_2[1]],[0,0,1]])
    x2 = np.dot(T2,x2)

    # compute F with the normalized coordinates
    F = eight_point(x1,x2)

    # reverse normalization
    F = np.dot(T1.T,np.dot(F,T2))

    return F


def compute_epipole(F):
    '''
    This function computes the epipoles for a given fundamental matrix
    and corner point correspondences
    input:
    F: Fundamental matrix
    output:
    e1: corresponding epipole in image 1
    e2: epipole in image2
    '''
    ### YOUR CODE HERE
    w, v = np.linalg.eig(F)
    e2 = v[:, np.argmin(np.abs(w))]

    w, v = np.linalg.eig(F.T)
    e1 = v[:, np.argmin(np.abs(w))]
    return e1, e2
    ### YOUR CODE ENDS

    return e1, e2



def plot_epipolar_lines(img1,img2, cor1, cor2):
    """Plot epipolar lines on image given image, corners

    Args:
        img1: Image 1.
        img2: Image 2.
        cor1: Corners in homogeneous image coordinate in image 1 (3xn)
        cor2: Corners in homogeneous image coordinate in image 2 (3xn)

    """
    F = fundamental_matrix(cor1, cor2)
    e1, e2 = compute_epipole(F)
    ### YOUR CODE HERE
    def draw_line(ax, x1, x2, mn, mx):
        k = (x2[1] - x1[1]) / (x2[0] - x1[0])
        b = x1[1] - k * x1[0]
        ax.plot([mn, mx], [k*mn+b, k*mx+b], color='blue')

    fig = plt.figure(figsize=(8, 8))
    ax = fig.add_subplot(111)
    ax.imshow(img1, cmap='gray')
    for c in cor1.T:
        draw_line(ax, e1[:2] / e1[2], c[:2], 0, img1.shape[1])
    ax.scatter(cor1[0], cor1[1])
    plt.xlim([0, img1.shape[1]])
    plt.ylim([img1.shape[0],0])
    plt.show()

    fig = plt.figure(figsize=(8, 8))
    ax = fig.add_subplot(111)
    ax.imshow(img2, cmap='gray')
    for c in cor2.T:
        draw_line(ax, e2[:2] / e2[2], c[:2], 0, img2.shape[1])
    ax.scatter(cor2[0], cor2[1])
    plt.xlim([0, img2.shape[1]])
    plt.ylim([img2.shape[0],0])
    plt.show()
    ### YOUR CODE ENDS
    return


plot_epipolar_lines(warriors[0], warriors[1], cor1, cor2)
```


0
200
400
600
800
1000
1200
1400
1600
0
200
400
600
800
1000
1200
1400
1600


0
200
400
600
800
1000
1200
1400
1600
0
200
400
600
800
1000
1200
1400
1600