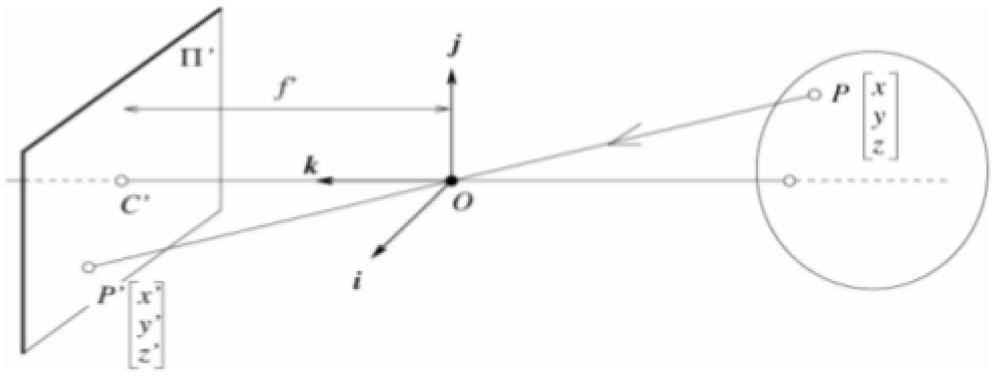
CSE 152 Homework 3

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 $-\infty \leq t \leq -1$.

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

With perspective projection: $Q1' = [\frac{7+8t}{1+4t}f' \quad \frac{-3+2t}{1+4t}f' \quad f'] = [(2 + \frac{5}{1+4t})f' \quad (\frac{1}{2} + \frac{-7}{2(1+4t)})f' \quad f']$.

Endpoints of $Q1'$: Take t = -1, $[\frac{1}{3}f' \quad \frac{5}{3}f' \quad f']$. Take t = $-\infty$, $[2f' \quad \frac{1}{2}f' \quad f']$

$Q2' = [\frac{3+8t}{9+4t}f' \quad \frac{-5+2t}{9+4t}f' \quad f'] = [(2 + \frac{-15}{9+4t})f' \quad (\frac{1}{2} + \frac{-19}{2(9+4t)})f' \quad f']$.

Endpoints of $Q2'$: Take t = -1, by similar triangles, we get $[-f' \quad -\frac{7}{5}f' \quad f']$. However, the start point is at the left of the camera, then this would result in infinity. Take t = $-\infty$, $[2f' \quad \frac{1}{2}f' \quad f']$

Therefore, Vanishing point: $[2f' \quad \frac{1}{2}f' \quad f']$

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]

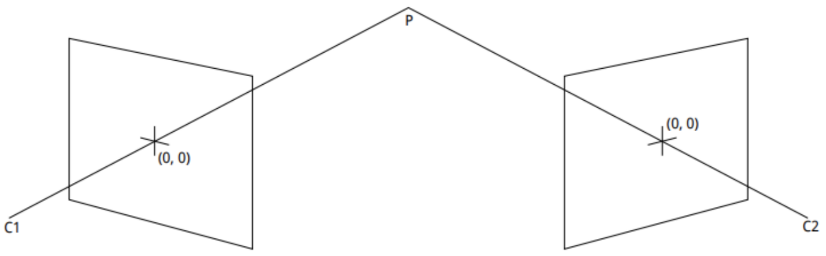


Figure 1: $C1$ and $C2$ are the optical centers. The principal axes intersect at point P.

Your proof here

Since $(0, 0)$ concides with the principal point, $x_1 = C1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, x_2 = C2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.

$$x_1^T F x_2 = 0 \quad => \quad \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} F \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

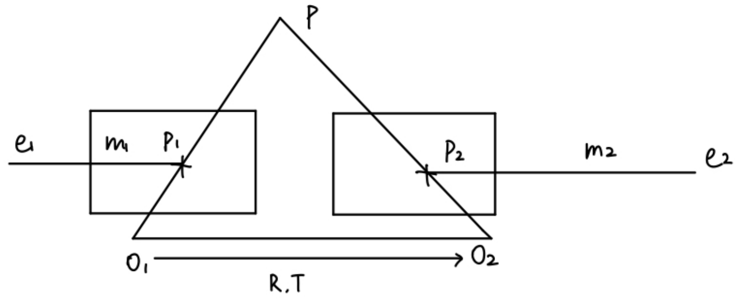
$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$F_{33} = 0$$

- 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



Proof: Since only a pure translation parallel to the image plane, then $R=I$, and $T=[T_x, T_y, 0]^T$.

Let $P_1=[u_1, v_1, 1]^T$, $P_2=[u_2, v_2, 1]^T$.

P_2 in first camera reference system is $= RP_2+T = P_2+T$

$T\times[P_2+T] = T\times P_2$ is perpendicular to epipolar plane.

then $P_1^T\cdot[T\times P_2]=0 \Rightarrow P_1^T[T_\times]P_2=0$, $E=[T_\times]$.

$m_1 = EP_2$ is the epipolar line associated with P_2.

$$= [T_\times]P_2 = \begin{bmatrix} 0 & 0 & T_y \\ 0 & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = \begin{bmatrix} T_y \\ -T_x \\ -T_y u_2 + T_x v_2 \end{bmatrix}$$

Since m_1 is in image plane 1, then the equation of m_1 is: $T_y a - T_x b + (-T_y u_2 + T_x v_2) = 0$

Similarly, $m_2 = E^T P_1$ is the epipolar line associated with P_1.

$= [T_\times]^T P_1 = -[T_\times]P_1$ (By property of skew-symmetric matrix)

$$= \begin{bmatrix} -T_y \\ T_x \\ T_y u_1 - T_x v_1 \end{bmatrix}$$

Since m_2 is in image plane 2, then the equation of m_2 is: $-T_y a + T_x b + (T_y u_1 - T_x v_1) = 0$

Thus, for any arbitrary P_1, P_2, the epipolar lines can be written as: $T_y a - T_x b + c = 0$

Therefore, epipolar lines in the two cameras are parallel.

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).

3.1 Set up

Let's first plot the images and points.

In [1]:

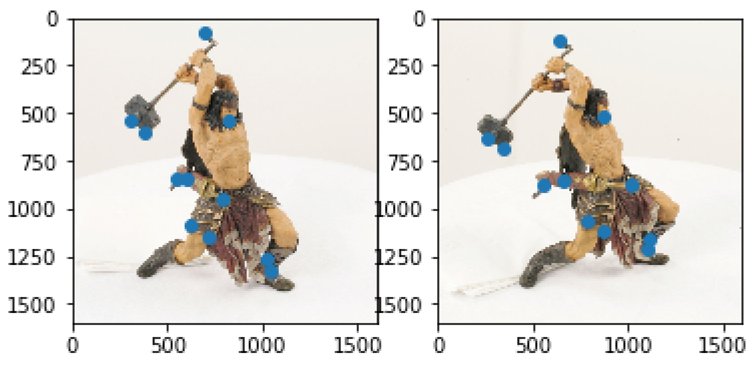
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage.filters import gaussian_filter, convolve
import scipy
import warnings
from skimage.io import *
warnings.filterwarnings('ignore')
```

In [2]:

```
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])
```

Out[2]:

```
<matplotlib.collections.PathCollection at 0x10b5a94d0>
```



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

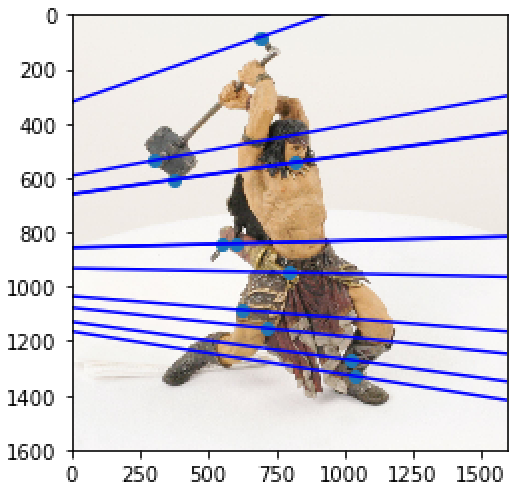
In this question, you need to implement 3 methods: `eight_point` , `fundamental_matrix` , `compute_epipole` .

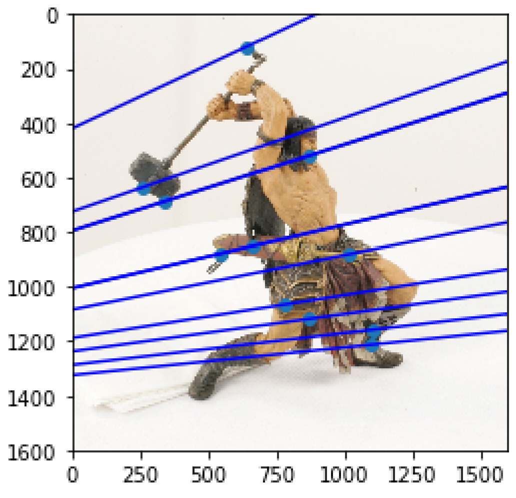
1. In `eight_point` , you should use 8-point algorithm to estimate the fundamental matrix. However, the provided points are larger than 8, so it becomes an over-determinated linear system. So you should get a least square solution for the fundamental matrix F, and ensure $rank(F) = 2$ by set the last 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.

In [5]:

```
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.")

    F = None

    # build matrix for equations
    # x2^TFx1 = 0
    A = np.zeros((n,9))
    A[:,0] = x2[0] * x1[0]
    A[:,1] = x2[0] * x1[1]
    A[:,2] = x2[0]
    A[:,3] = x2[1] * x1[0]
    A[:,4] = x2[1] * x1[1]
    A[:,5] = x2[1]
    A[:,6] = x1[0]
    A[:,7] = x1[1]
    A[:,8] = np.ones(n)
#     print(A)
    # compute linear least square solution
    U, S, VT = np.linalg.svd(A)
    f = VT[-1].reshape((3,3))

    # constrain F: make rank 2 by zeroing out last singular value
    U, S, VT = np.linalg.svd(f)
    S[2] = 0
    sigma = np.diag(S)
    F = U @ sigma @ VT

    return F


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
    mean1 = np.mean(x1,axis = 1)
    dist1 = [np.sqrt((x1[:,i] - mean1)@(x1[:,i] - mean1)) for i in range(n)]
    mean_dist1 = np.mean(dist1)
    diag1 = [np.sqrt(2)/mean_dist1,np.sqrt(2)/mean_dist1,1]
    T1 = np.diag(diag1)
    T1[:2,2] = (-np.sqrt(2)*mean1/mean_dist1)[:2]
#     print(T1)

    mean2 = np.mean(x2,axis = 1)
    dist2 = [np.sqrt((x2[:,i] - mean2)@(x2[:,i] - mean2)) for i in range(n)]
    mean_dist2 = np.mean(dist2)
```

```python
    diag2 = [np.sqrt(2)/mean_dist2,np.sqrt(2)/mean_dist2,1]
    T2 = np.diag(diag2)
    T2[:2,2] = (-np.sqrt(2)*mean2/mean_dist2)[:2]
#     print(T2)

    x1 = T1 @ x1
#     print(np.mean(x1,axis=1))
#     print(np.mean([x1[0][i]**2 + x1[1][i]**2 for i in range(11)]))
    x2 = T2 @ x2
#     print(np.mean(x2,axis=1))
#     print(np.mean([x2[0][i]**2 + x2[1][i]**2 for i in range(11)]))
    # compute F with the normalized coordinates
    F = eight_point(x1,x2)

    # reverse normalization
    F = T2.T @ F @ T1

    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
    # Fe1 = 0
    U, S, VT = np.linalg.svd(F)
    e1 = VT[-1]

    #F^Te2 = 0
    U, S, VT = np.linalg.svd(F.T)
    e2 = VT[-1]

    e1 = e1/e1[2]
    e2 = e2/e2[2]
    ### 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)
#     print(e1)
#     print(e2)
    ### YOUR CODE HERE
    plt.imshow(warriors[0])
    plt.scatter(cor1[0], cor1[1])
    for i in range(11):
        slope = (cor1[1][i] - e1[1])/(cor1[0][i] - e1[0])
        x = np.array([0,1600])
        y = slope*(x-e1[0]) + e1[1]
        plt.plot(x,y,"-r", c="b")
    plt.xlim(left = 0, right = 1600)
    plt.ylim(top = 0)
    plt.show()

    plt.imshow(warriors[1])
    plt.scatter(cor2[0], cor2[1])
    for i in range(11):
        slope = (cor2[1][i] - e2[1])/(cor2[0][i] - e2[0])
        x = np.array([0,1600])
        y = slope*(x-e2[0]) + e2[1]
        plt.plot(x,y,"-r", c="b")
    plt.xlim(left = 0, right = 1600)
    plt.ylim(top = 0)
    plt.show()
    ### YOUR CODE ENDS
    return


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





In [4]:

```
warriors[0].shape
```

Out[4]:

```
(1600, 1600, 3)
```

In [ ]:

```

```