CS-E4850 Computer Vision, Answers to Exercise Round 12

Yangzhe Kong, Student number: 765756

December 3, 2019

Exercise 2. Epipolar geometry.

Here we use the position of the camera 2 as our origin. Let's first write the equation (1) out using the notation from $\mathbf{x}$ and $\mathbf{x}'$:

$$\vec{O'O} = \mathbf{t} = (t_1, t_2, t_3)^\top$$

$$\vec{O'p'} = \mathbf{x}'$$
$$\vec{Op} = R\mathbf{x}$$

Thus we can derive that:

$$\vec{O'p'} \cdot (\vec{O'O} \times \vec{Op}) = \mathbf{x}' \cdot (\mathbf{t} \times R\mathbf{x}) = 0$$

where $\mathbf{t} \times R\mathbf{x}$ can be written as $[\mathbf{t}]_\times R\mathbf{x}$ and $\mathbf{t} \times \mathbf{t} = 0$. Therefore finally we get:

$$\mathbf{x}'^\top [\mathbf{t}]_\times R\mathbf{x} = 0$$

where $[\mathbf{t}]_\times R$ is the essential matrix E.

Exercise 3. Stereo vision.

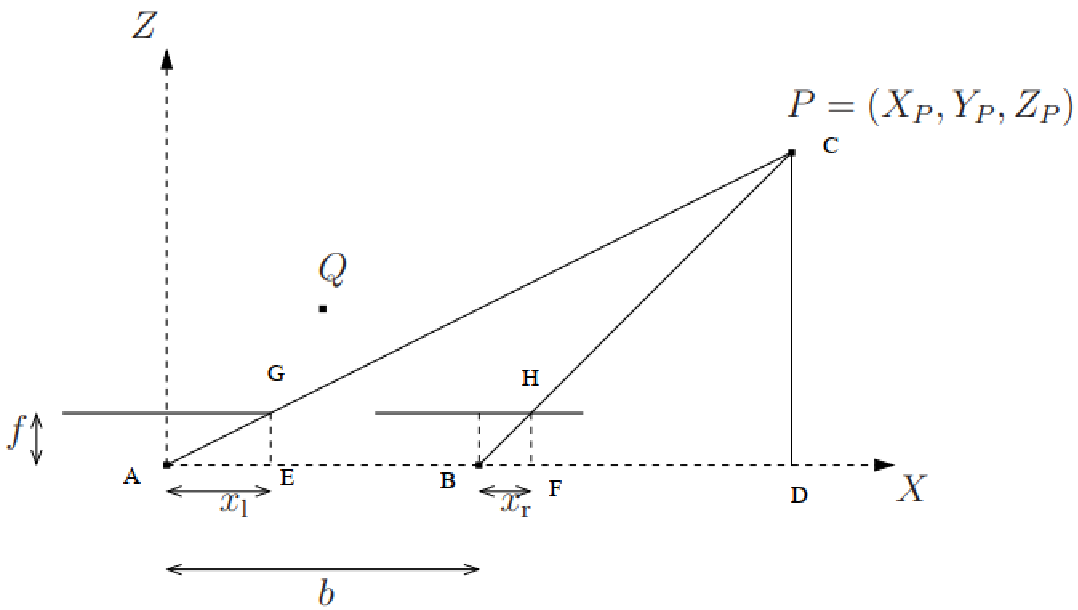

Figure 1: : Top view of a stereo configuration where two pinhole cameras are placed side by side

a) We have 2 pairs of similar triangles in Figure 1: ΔAGE and ΔADC, ΔBHF and ΔBDC. Let's assume that $DF = a$, thus we can have the following formulas:

$$\frac{f}{x_l} = \frac{Z_p}{b + x_r + a}$$

$$\frac{f}{x_r} = \frac{Z_p}{x_r + a}$$

$$x_l = 1 + x_r$$

Then we can get:

$$\frac{1}{x_l} = \frac{Z_p}{6 + x_r + a}$$

$$\frac{1}{x_r} = \frac{Z_p}{x_r + a}$$

$$x_l = 1 + x_r$$

Finally we can substitute x_l with $x_r + 1$ and we can derive that:

$$1 + x_r = \frac{6 + x_r + a}{Z_p}$$

$$x_r = \frac{x_r + a}{Z_p}$$

Therefore, we get that $Z_p = 6cm$

b) Actually we can use the similar method to derive that in the general case $\frac{d}{f} = \frac{b}{Z_p}$ (because $\frac{f}{x_l} = \frac{Z_p}{b+x_r+a}$, $\frac{f}{x_r} = \frac{Z_p}{x_r+a}$ and $|x_l - x_r| = d$), and therefore $Z_p = \frac{bf}{d}$. Thus, if d is below than 1 pixel, we can say that $0 \leq d < 0.01mm$. So Z_p should fall into the range $(100bf, \infty)mm$

c) Q on the left image plane is:

$$q_l = P_r Q = (1, 0, 1)$$

Then the essential matrix is:

$$E = [t]_\times R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 6 \\ 0 & -6 & 0 \end{bmatrix}$$

So then the corresponding epipolar line l' with q_l is $Eq_l = (0, 1, 0)^\top$. And since $q_r = (-1, 0, 1)^\top$ should be on the line l' too, thus we have the formula for the coeffients a, b, c, d of l':

$$a = 0$$

$$b = 1$$

$$c = 0$$

$$d = 0$$

And also because l' is on the second image plane, thus l' should be $y = 0, z = 0$

Exercise 4. Fundamental matrix estimation.

The python code for the two functions are shown below.

```python
def estimateF(x1,x2):
    # Return the fundamental matrix F (3 by 3), based on two sets of homogeneous 2D points x1 and x2.
    # Input: x1,x2 numpy ndarray (3 by N) containing matching 2D homogeneous points.
    # Output: F numpy ndarray (3 by 3) containing the fundamental matrix.
    n = x1.shape[1]
    A = np.zeros((n,9))
    for i in range(n):
```

```
        u,v,u_,v_ = x1[0,i]/x1[2,i],x1[1,i]/x1[2,i],x2[0,i]/x2[2,i],x2[1,i]/x2[2,i]
        A[i] = [u_*u,u_*v,u_,v_*u,v_*v,v_,u,v,1]
    # compute linear least square solution
    U,S,V = np.linalg.svd(A)
    F = V[-1].reshape(3,3)
    print(F)
    # rank 2 constrain on F
    U,S,V = np.linalg.svd(F)
    S[2] = 0
    F = (U*S)@V
    print(F)
    return F

def estimateFnorm(x1,x2):
    # Return the fundamental matrix F (3 by 3), based on two sets of homogeneous 2D points x1 and x2.
    # Input: x1,x2 numpy ndarray (3 by N) containing matching 2D homogeneous points.
    # Output: F numpy ndarray (3 by 3) containing the fundamental matrix based on normalized homogeneous points.
    n = x1.shape[1]

    # 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 = estimateF(x1,x2)

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

```
    return F
```