Computing Homography: Guidelines

1 Overview

The main part of the homework is to calculate homography transformation, mapping from one viewpoint to another. We will formulate it as a least squares estimation problem.

For one pair of correspondent points $[x,\ y,\ 1]^T$ and $[x', y', 1]^T$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \propto \mathbf{Hx} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\Rightarrow x' = \frac{h_{11}x+h_{12}y+h_{13}}{h_{31}x+h_{32}y+h_{33}} \text{ and } y' = \frac{h_{21}x+h_{22}y+h_{23}}{h_{31}x+h_{32}y+h_{33}}$$

And these can be written as equations for parameters of H (given $h_{33} = 1$):

$$\begin{aligned} h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' = x' \\ h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' = y' \end{aligned}$$

or in matrix form:

$$\mathrm{A}\mathbf{h}=\mathbf{b}$$

$$A = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -xx' & -yx' \\ 0 & 0 & 0 & x & y & 1 & -xy' & -yy' \end{bmatrix}, b = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{1}$$

$$h = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} \end{bmatrix}^{\mathrm{T}} \tag{2}$$

Similarly, with n pairs of correspondences, we can create a system of 2n equations for $\mathbf{h}$: A$\mathbf{h}$ = b, where every two rows of A and $\mathbf{b}$ will represent the relation of each pair. A will be a matrix of size 2n $\times$ 8 and b will be a vector of size 2n $\times$ 1. In order to solve for $\mathbf{h}$, we will need at least 4 pairs.

- For computational stability, we often want to collect more than 4 pairs and have an over-determined system where the number of equations is greater than the number of variables (2n x 8).
- In this case, $\mathbf{h}$ can be found by minimizing the squared error $||A\mathbf{h} - \mathbf{b}||_2^2$, $\mathbf{h}$ is then called the least squares solution for the over-determined system.
- This might sound complicated, but to find $\mathbf{h}$ in Python, we can use **numpy.linalg.solve** or **numpy.linalg.lstsq**.

2 Implementation

In the homework you are asked to do the following:

1. Identify pairs of correspondent points in two images. At least 4 pairs are needed for the algorithm but it's recommended to use 8 or even more for better results. In this step you might want to explore function **ginput** in the **mathplotlib.pyplot** library and might as well save those points to a file so that you can reuse later. **Please note that ginput cannot be used in Pycharm or Google Colab**. You can take the following script as a reference, and modify it based on your needs.

```
from matplotlib import cm
from matplotlib import pyplot as plt
import numpy as np
from skimage import io

img1 = io.imread('Left.jpg')
img2 = io.imread('Right.jpg')

plt.imshow(img1)
# Depending on the environment, a plt.show() might be needed to properly
# display the image
points_left = plt.ginput(8)

plt.imshow(img2)
points_right = plt.ginput(8)

#saving data into file for future use
np.save('right.npy', points_right)
np.save('left.npy', points_left)
```

2. Write a function, *homography*, for example, to calculate homography transformation from one image to the other using the least squares method. The function should take the corresponding pairs as input and output the transformation matrix **H**. Your job is basically putting those pairs' coordinates into the matrix A and vector **b** so that Python can do the rest. The function might look something like this.

```
import numpy as np
from scipy.linalg import solve

def homography ( P1 , P2 ):
    # Your code here
    # ..............
    h = solve( A , b) #You can also use numpy.linalg.lstsq
    h = np.append(h,1)
```

```
    # Be aware of how reshape function works
    # and the maping between vector h and matrix H
    H = np.reshape (h , 3 , 3)
    return H
```

3. Finally, using the *homography* to do the transform from one view to the other. For this, you should look into function **cv2.warpPerspective** in Python. You might have a main script similar to this one.

```
import cv2
#Open Point data
points_left=np.load('left.npy')
points_right=np.load('right.npy')
#Solve for the homography using the function from before
H21 = homography(points_left , points_right)
# Apply the transformation to the image
im21 = cv2.warpPerspective(img2, H21)

plt.figure
plt.imshow(img21)

plt.figure
plt.imshow(img1)
```

4. You have repeat this for 2 pairs of images. For each pair, you need to calculate transformation from one image to the other. So totally you will have 4 transformations to calculate. During the process, remember to save necessary information needed for your report.