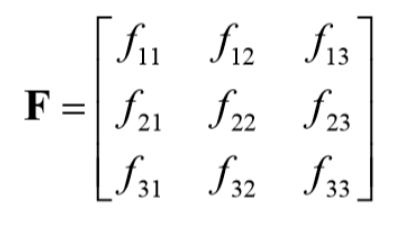
**Computer Vision Homwork2 Report**

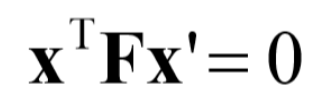
111062569 葉政賢

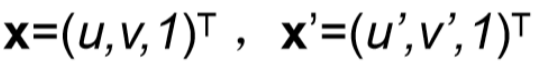
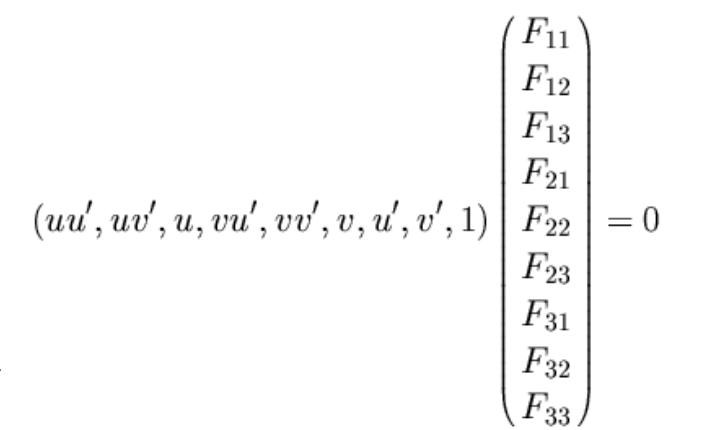
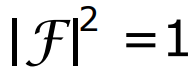
1. In this problem, you will implement both the linear least-squares version of the eight-point algorithm and its normalized version to estimate the fundamental matrices. You will implement the methods and complete the following:

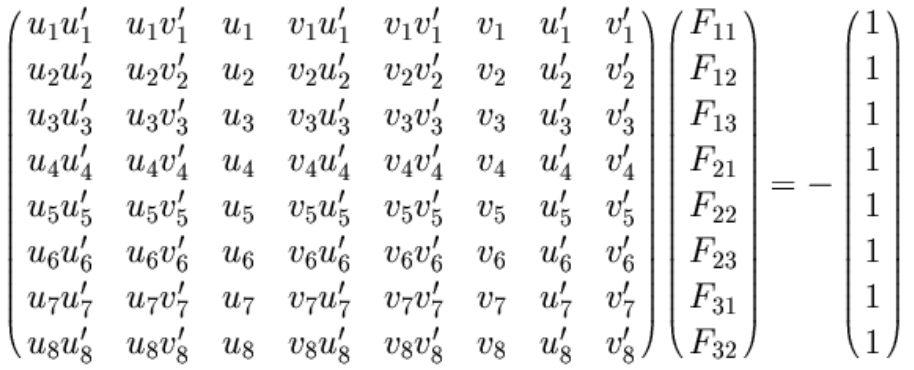
(a) Implement the linear least-squares eight-point algorithm and report the returned fundamental matrix. Remember to enforce the rank-two constraint for the fundamental matrix via singular value decomposition.

***Theorem***

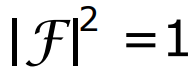
Funfamenta; matrix 定義為 ，能夠使得兩幅圖片的點滿足：

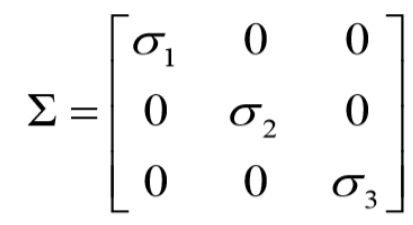
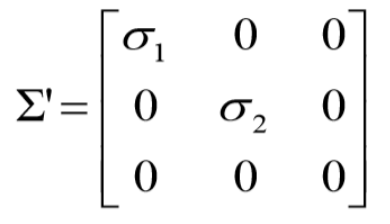


假設，則會使得。此時，將設為1，並滿足的constraint，則



***Implement***

首先求解分解求得參數解，滿足。

然而，上述求解並不能滿足 rank2的條件，因此在通過一次SVD，使得，將改成，最後即為所求。

***-- Pseudo code --***

>. M = ,1

>. U,D,V = SVD(M)

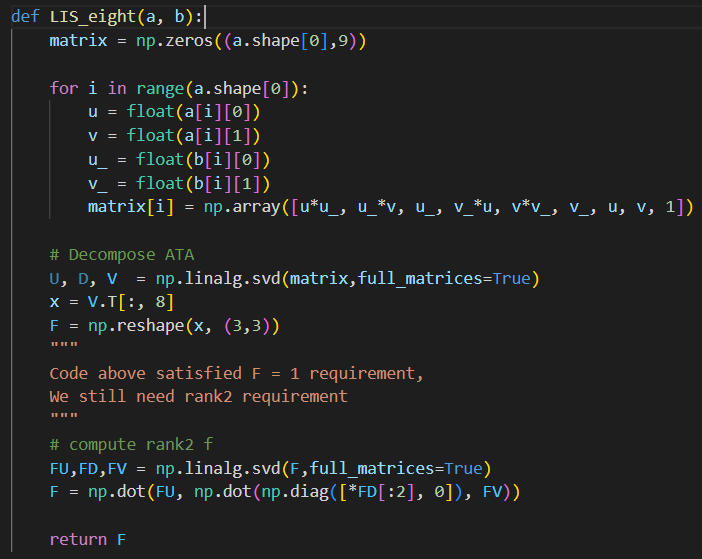
>. F = reshape(V(:9), 3,3)

>. FU, FD, FV = SVD(F)

>. FD(3,3) = 0

>. F = FU \* FD \* FV

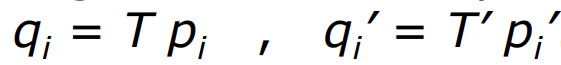
***-- Code Implement --***



(b) Implement the normalized eight-point algorithm and report the returned fundamental matrix. Remember to enforce the rank-two constraint for the fundamental matrix via singular value decomposition.

***Theorem***

1. Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels



1. Use the eight-point algorithm to compute F from the points and .
2. Enforce the rank-2 constraint
3. Output

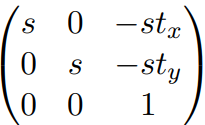
***Implement***

***-- Pseudo --***

>. center = uv – mean(uv)

>. ***# To satisfy the criteria that the average distance of a point p from the origin is equal to √ 2***

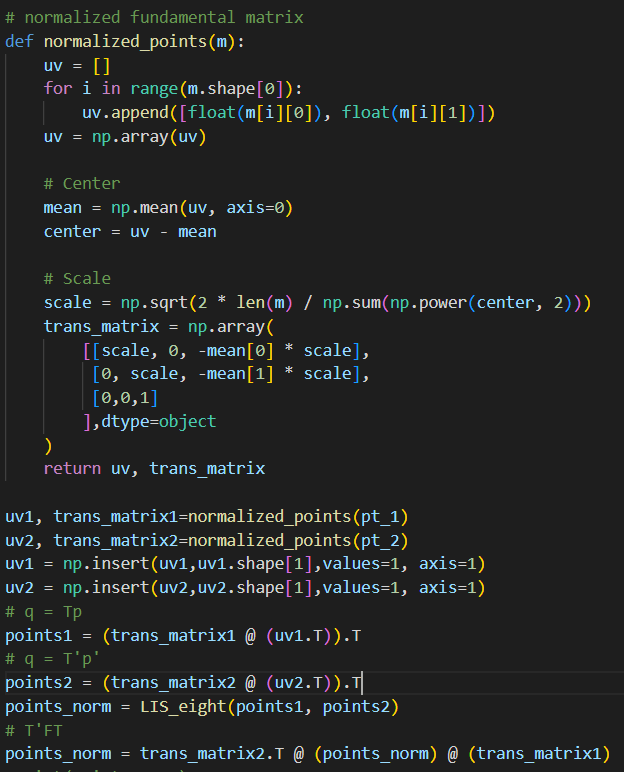
>. Distance =

>. Transform matrix = 

>. LTS\_eight()

>.

***-- Code Implement --***



3. Plot the epipolar lines for the given point correspondences determined by the fundamental matrices computed from (a) and (b). Determine the accuracy of the fundamental matrices by computing the average distance between the feature points and their corresponding epipolar lines.

***Theorem***

畫出epipolar lines，

將 fundamental matrix \* points會得到epipolar lines係數。

對於未normalized 的fundamental matrix所畫出的線會超出圖片本身。原則上先固定x 為 [0, 圖片寬度]去求y值，再畫線。若 x = 0時，y會超出圖片，則改變以y = 0或圖片高度去求x值畫線。

計算距離方面，

將各點帶入 Ax + By + C 的平面中，再去與 相除得到與epipolar lines的距離。最後將距離除以總points得到平均距離。

***Implement***

***-- Pseudo --***

>. # plot

>. ln = f.T.dot(pt2.T)

>. A,B,C = ln

>. # when y as 0，x = - (C/A)

   # when y = height, x = -(Bw + C / A)

   # when x as width, y = - (Aw + C / B)

   # when x as 0, y = - (C / B)

>. If (-C / B <0):

>. Plot((-C/A, width), (0, -(C + A\*width) / B))

>. elif(-C / B > height):

>. Plot((-C/A, width), (height, -(C + A\*width) / B))

>. else:

>. Plot((0,width), (-C/B, -(C + A\*width) / B)

>. # plot normalized

>. ln = f.T.dot(pt2.T)

>. A,B,C = ln

>. # when x as width, y = - (Aw + C / B)

   # when x as 0, y = - (C / B)

>. Plot((0,width), (-C/B, -(C + A\*width) / B)

>. # Calculate Distance

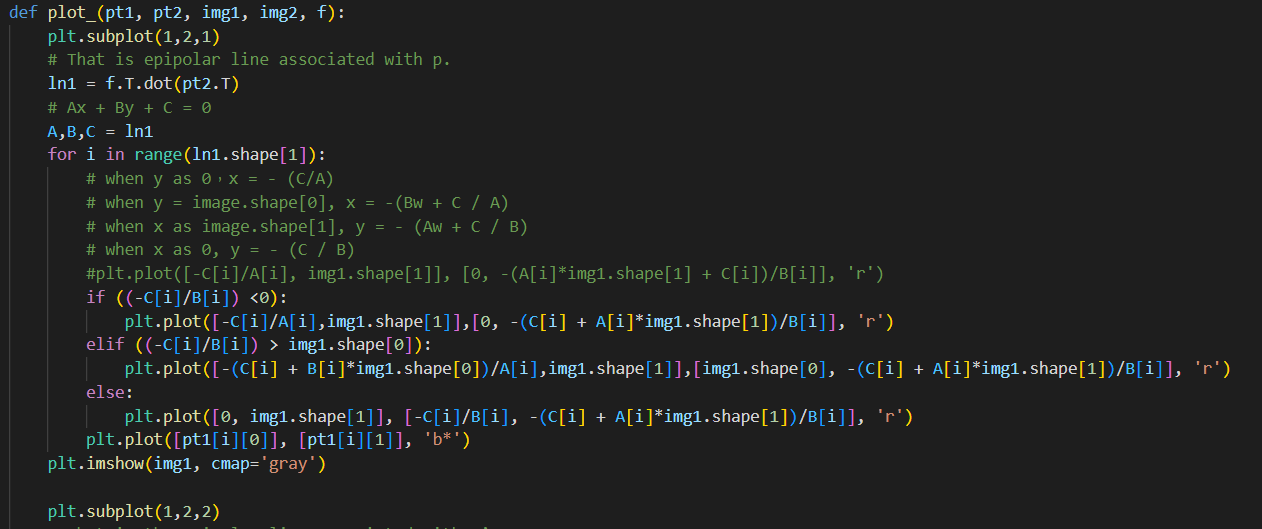
>. ln = f.T.dot(pt2.T)

>. dist = abc(Ax + By + C) / sqrt(A\*\*2 + B\*\*2)

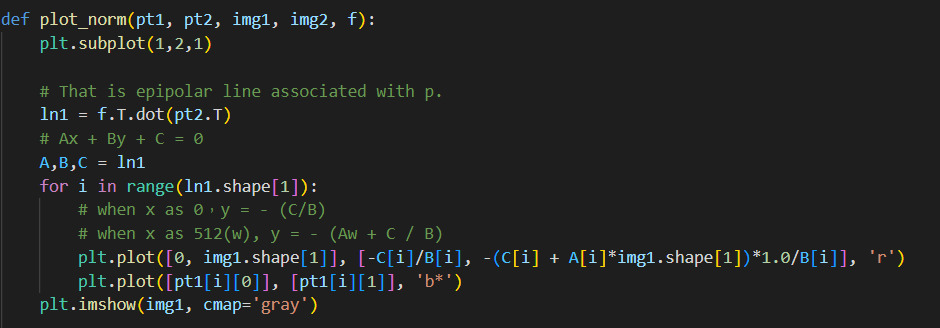
>. dist / points.number

***-- Code Implement --***

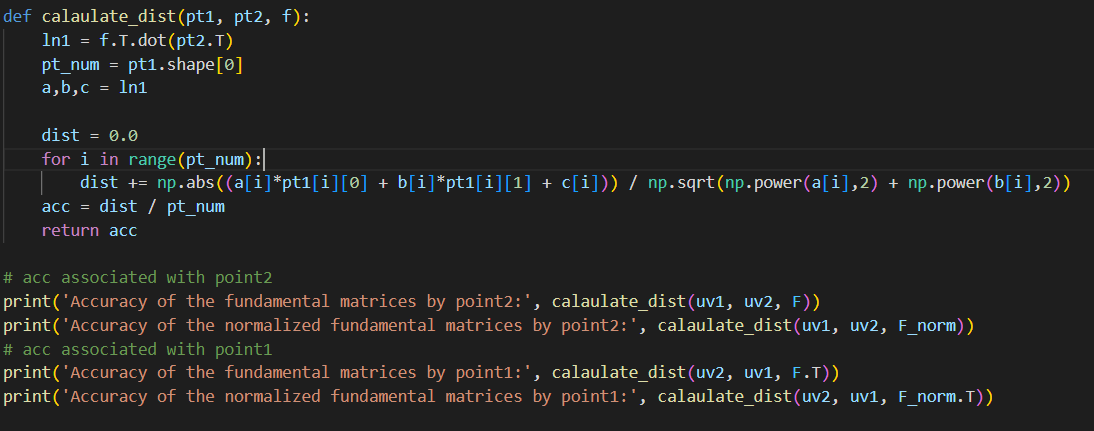
Plot



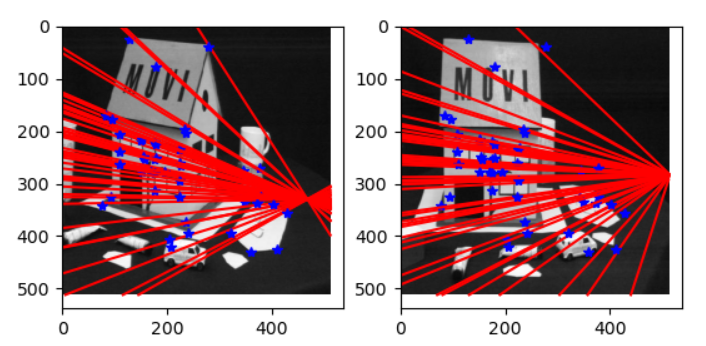
Plot Normalized

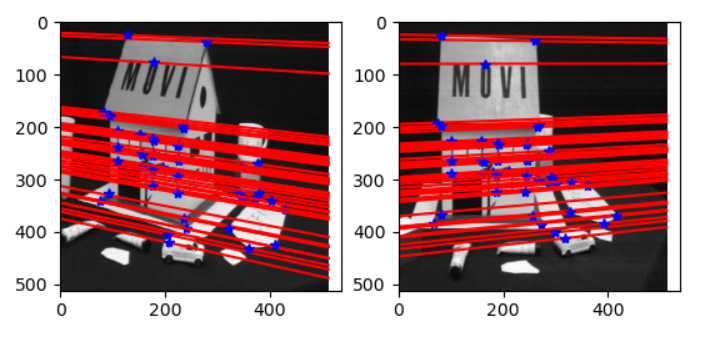


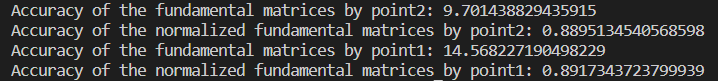
Calculate Distance



***-- Result --***





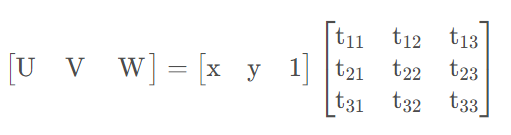


2. You need to determine a homograpgy transformation for plan-to-plane transformation. The homography transformation is determined by a set of point correspondences between the source image and the target image.

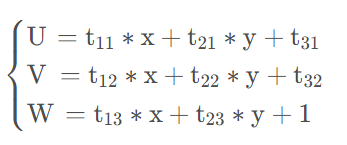
(a) Implement a function that estimates the homography matrix H that maps a set of interest points to a new set of interest points.

***Theorem***

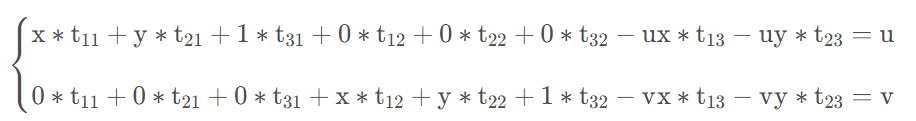
兩個座標座標的轉換矩陣為

，但是將其化為方程組的話，會陰沒有常數項難以求解。因此我們將 d改寫為1。

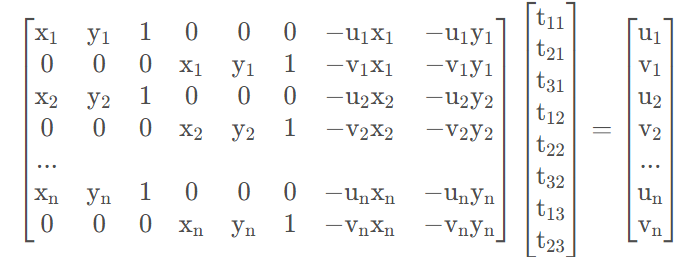
方程組將變為：



整理可得：



因此，寫成線性方程組將為如下：



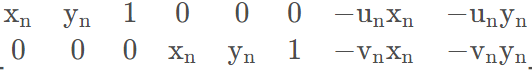
因下一題為將台達館轉至正面，因此此時我直接將台達館正面區域作為 interest points，轉至512 X 512的圖片，也就是new set interest points。

台達館正面座標依順時針依序為： [425,330],[400,800],[900,1000],[880,0]

新座標為：[0,0],[0,512],[512,512],[512,0]

***Implement***

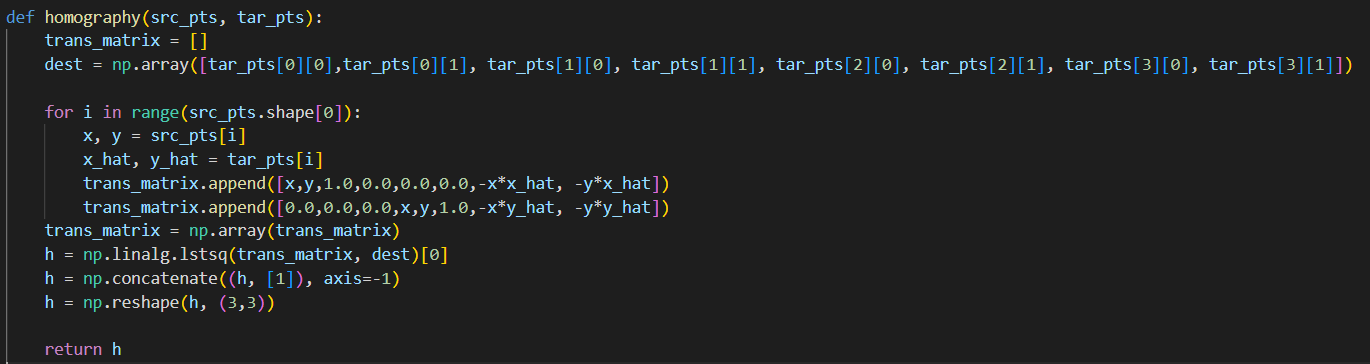
***-- Pseudo --***

>. M = 

>. H = Least Square(M, destination\_points)

>. H = reshape(H, 3,3)

***-- Code Implement --***



(b) Specify a set of point correspondences for the source image of the Delta building and the target one. Compute the 3X3 homography matrix to rectify the front building of the Delta building image. The rectification is to make the new image plane parallel to the front building as best as possible. Please select four corresponding straight lines to compute the homograph matrix.

***Theorem***

Homography matrix 於上一題已提及，之後進行backward warping and bilinear interpolation。

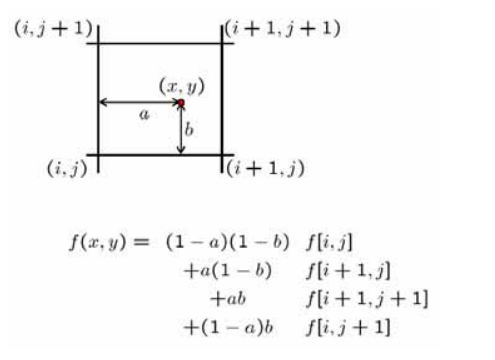
Backward warping與 forward mapping不同，反而是從destination 座標去反求 source 座標位置，再將其複製 pixel過來 destination 上。

因此，需先destination points 乘上 Homography matrix 反矩陣求得source points。

然而透過反矩陣求得的座標為浮點數，此時利用 bilinear interpolation 去得到鄰近4的點的pixel 值再依比例綜合。

具體如下：

先求得上下兩邊x軸的比例，再依y軸比例分配各pixel 值。



***Implement***

***-- Pseudo –***

>. For (x,y): u,v =

>. # bilinear interpolation

>. a = ceil(u)

>. b = floor(u)

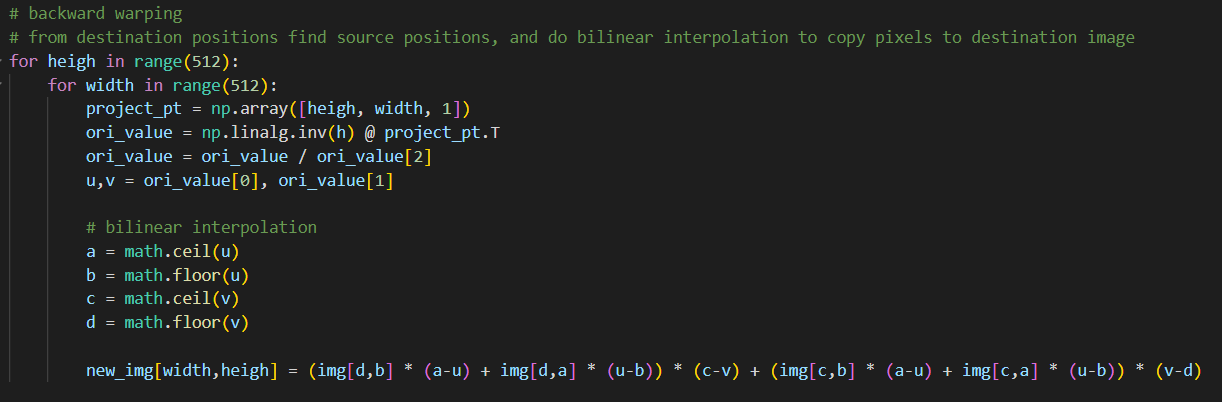
>. c = ceil(v)

>. d = floor(v)

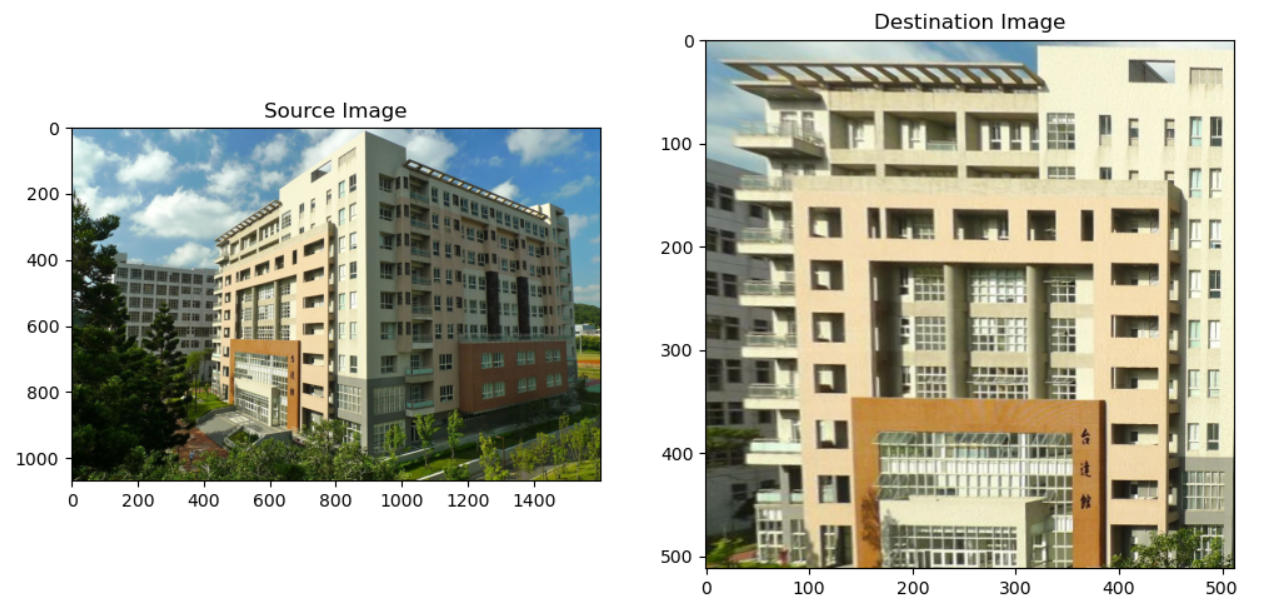
>. dst[x,y] = [src[a,d] \* (u-b) + sec[b,d] \* (a-u)] \* (c-v) + /

>. [src[a,c] \* (u-b) + sec[b,c] \* (a-u)] \* (v-d)

***-- Code Implement –***



***-- Result --***





原圖越傾斜的部分轉換過來會越模糊。

***Another Experiment***

實驗使用forward mapping與 取floor的投射:

