**Computer Vision HW3 Report**

Student ID: R11921103

Name: 張銘軒

**Part1: Homography estimation**

**• Paste your warped canvas**

**Ans:**

|  |
| --- |
|  |

**Part2: Marker-Based Planar AR**

**• Paste the function code solve\_homography(u, v) & warping( ) (both forward &  
backward)**

**Ans:**

|  |
| --- |
| solve\_homography(u, v) |
| def solve\_homography(u, v):  """  This function should return a 3-by-3 homography matrix,  u, v are N-by-2 matrices, representing N corresponding points for v = T(u)  :param u: N-by-2 source pixel location matrices  :param v: N-by-2 destination pixel location matrices  :return:  """  N = u.shape[0]  if v.shape[0] is not N:  print('u and v should have the same size')  return None  if N < 4:  print('At least 4 points should be given')  # TODO: 1.forming A  A = []  for i in range(N):  A.append([u[i][0], u[i][1], 1, 0, 0, 0, -u[i][0] \* v[i][0], -u[i][1] \* v[i][0], -v[i][0]])  A.append([0, 0, 0, u[i][0], u[i][1], 1, -u[i][0] \* v[i][1], -u[i][1] \* v[i][1], -v[i][1]])    # TODO: 2.solve H with A  res = np.linalg.svd(A, full\_matrices=True)  H = np.reshape(res[2][-1], (3,3))  return H |
| warping( ) |
| def warping(src, dst, H, ymin, ymax, xmin, xmax, direction='b'):  """  Perform forward/backward warpping without for loops. i.e.  for all pixels in src(xmin~xmax, ymin~ymax), warp to destination  (xmin=0,ymin=0) source destination  |--------| |------------------------|  | | | |  | | warp | |  forward warp | | ---------> | |  | | | |  |--------| |------------------------|  (xmax=w,ymax=h)  for all pixels in dst(xmin~xmax, ymin~ymax), sample from source  source destination  |--------| |------------------------|  | | | (xmin,ymin) |  | | warp | |--| |  backward warp | | <--------- | |\_\_| |  | | | (xmax,ymax)|  |--------| |------------------------|  :param src: source image  :param dst: destination output image  :param H:  :param ymin: lower vertical bound of the destination(source, if forward warp) pixel coordinate  :param ymax: upper vertical bound of the destination(source, if forward warp) pixel coordinate  :param xmin: lower horizontal bound of the destination(source, if forward warp) pixel coordinate  :param xmax: upper horizontal bound of the destination(source, if forward warp) pixel coordinate  :param direction: indicates backward warping or forward warping  :return: destination output image  """  h\_src, w\_src, ch = src.shape  h\_dst, w\_dst, ch = dst.shape  H\_inv = np.linalg.inv(H)  # TODO: 1.meshgrid the (x,y) coordinate pairs  x, y = np.meshgrid(range(xmin,xmax),range(ymin,ymax))  # TODO: 2.reshape the destination pixels as N x 3 homogeneous coordinate  before = np.array([x, y, np.ones(x.shape)]).reshape((3, -1))  if direction == 'b':  # TODO: 3.apply H\_inv to the destination pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)  after = np.matmul(H\_inv, before)  after\_x, after\_y = (after[:2,] / after[2]).astype(int)  # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of source image)  mask = np.logical\_and(np.logical\_and(after\_x >= 0, after\_x < w\_src), np.logical\_and(after\_y >= 0, after\_y < h\_src))  after\_x, after\_y = after\_x[mask], after\_y[mask]  # TODO: 5.sample the source image with the masked and reshaped transformed coordinates  sample = src[after\_y, after\_x]  # TODO: 6. assign to destination image with proper masking  dst[ymin:ymax, xmin:xmax][np.reshape(mask, (ymax-ymin, xmax-xmin))] = sample  elif direction == 'f':  # TODO: 3.apply H to the source pixels and retrieve (u,v) pixels, then reshape to (ymax-ymin),(xmax-xmin)  after = np.matmul(H, before)  after\_x, after\_y = (after[:2,] / after[2]).astype(int)  # TODO: 4.calculate the mask of the transformed coordinate (should not exceed the boundaries of destination image)  mask = np.logical\_and(np.logical\_and(after\_x >= 0, after\_x < w\_dst), np.logical\_and(after\_y >= 0, after\_y < h\_dst))  # TODO: 5.filter the valid coordinates using previous obtained mask  after\_x, after\_y = after\_x[mask], after\_y[mask]  # TODO: 6. assign to destination image using advanced array indicing  dst[after\_y, after\_x] = np.reshape(src, (-1, 3))  return dst |

**• Briefly introduce the interpolation method you use**

**Ans:**

我直接將像素值四捨五入，找到最接近的整數點取其值作為warping的值。

**Part3: Unwarp the secret**

**• Paste the 2 warped images**

|  |  |
| --- | --- |
| **output3\_1.png** | **output3\_2.png** |
|  |  |

**• Discuss the difference between 2 source images, are the warped results the same or  
different? If the results are the same, explain why. If the results are different, explain why.**

**Ans:**

Warped results的照片很明顯是不同的。3\_1QR Code的紋路比較清楚而3\_2比較模糊，這是因為他們的原始照片中，BL\_secret2畫面拍攝起來較BL\_secret1扭曲許多，使得在warping回來的時後效果較差。但透過手機的QR Code掃描器仍然可以從兩張照片獲得一樣的網址。網址為：<http://media.ee.ntu.edu.tw/courses/cv/21S/>

**Part4: Panorama**

**• Paste your stitched panorama**

**  
• Can all consecutive images be stitched into a panorama? If yes, explain your reason. If not, explain under what conditions will result in a failure?**

**Ans:**

並不是所有consecutive images都可以stitch into a panorama，這些連續的照片必須有重複的部分讓程式辨識出來，若這些照片彼此毫無交界之處，或是重複之處少到無法將兩張照片連在一起，是無法stitch into a panorama的。