3D Vision – Assignment 3

魏家博 (Chia-Po Wei)

Department of Electrical Engineering

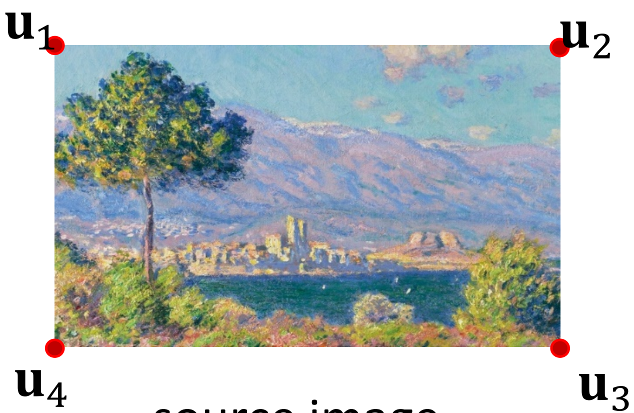
National Sun Yat-sen University

[TODO 1] Implement the function `calc_homography(pts_src, pts_dst)`

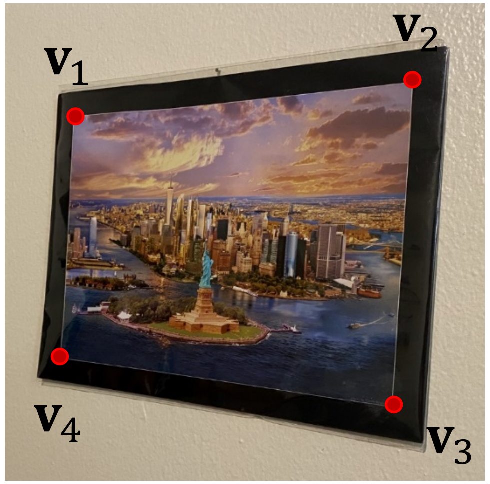
- `pts_src` and `pts_dst` are numpy arrays both of shape (4,2).
- This function returns the homography matrix H of shape (3,3).
- Hint:
    - Implement the DLT algorithm presented in page 11 of 06 Homography.pdf.
    - Use `np.linalg.svd` to perform singular value decomposition.
    - The matrix A in page 11 should be of `dtype=np.float64`. If the dtype of A is `np.int32`, the results of SVD might not be accurate.

[TODO 2] Project the source image on the frame formed by v1, v2, v3, and v4 in the target image.

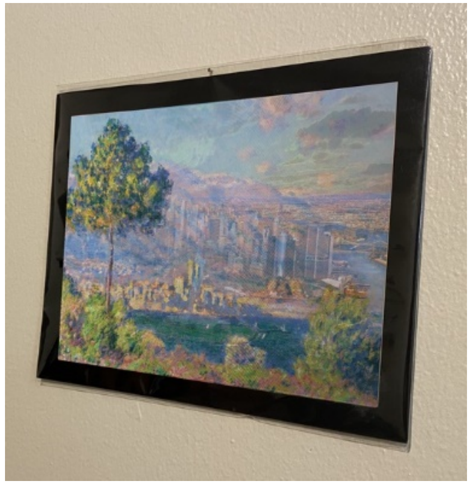
- The projection result is shown below.
- In this task, you need to use the direct projective transformation H mapping from source points (u1, u2, u3, u4) to target points (v1, v2, v3,v3).
- The algorithm is given in the next slide.



source image



target image



projection result

[TODO 2]

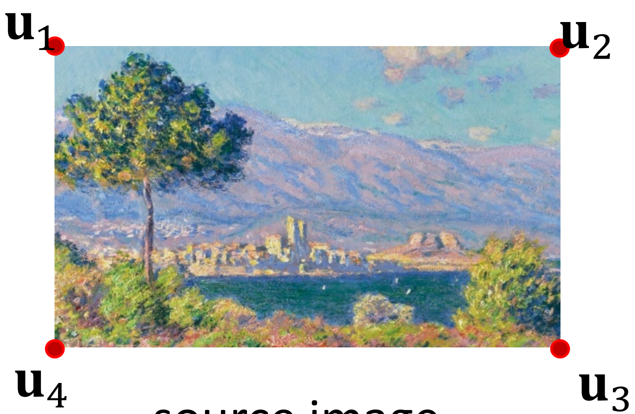
- Let (`ht_src, wid_src`) and (`ht_dst, wid_dst`) be the shapes of source and target images, respectively.

```
H = calc_homography(pts_src, pts_dst)
for j in range(ht_src):
    for i in range(wid_src):
        Use the homography matrix H to project $(i, j)$, and denote the result by $(x, y)$.
        if y < ht_dst and x < wid_dst:
            Copy the pixel of the source image at $(i, j)$ to the pixel of the target image at $(x, y)$.
```

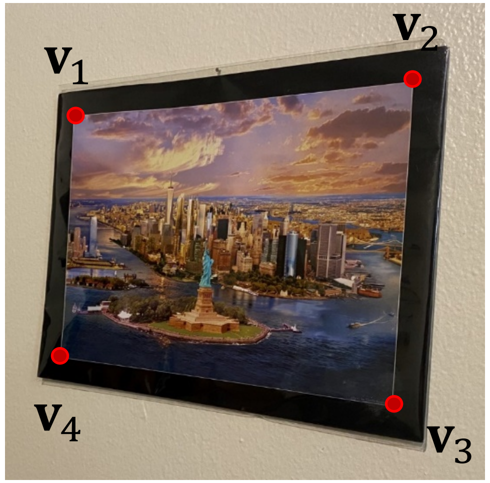
- Remark: The pixel at (i, j) means that the pixel is located at the ith column and jth row of an image.

[TODO 3] Project the source image on the frame formed by v1, v2, v3, and v4 in the target image.

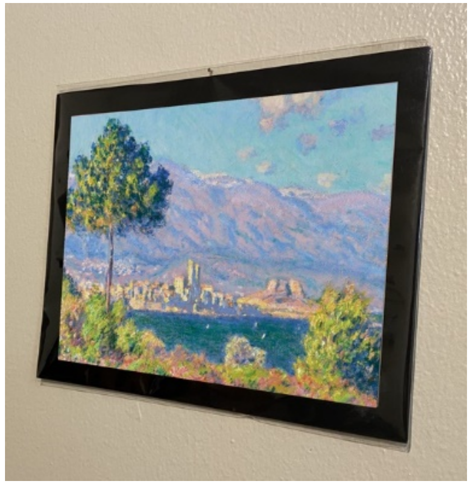
- The projection result is shown below.
- In this task, you need to use the inverse projective transformation H^{-1} mapping from target points (v1, v2, v3,v3) to source points (u1, u2, u3, u4).



source image



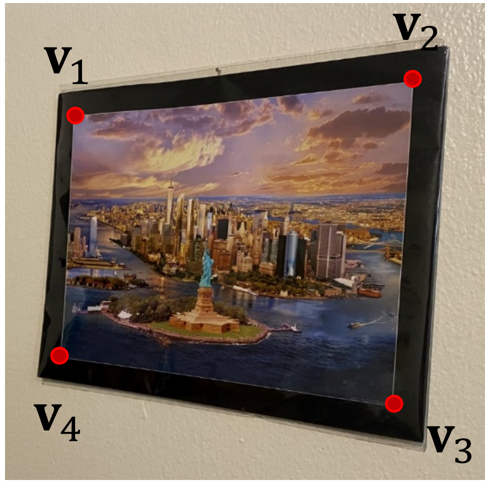
target image



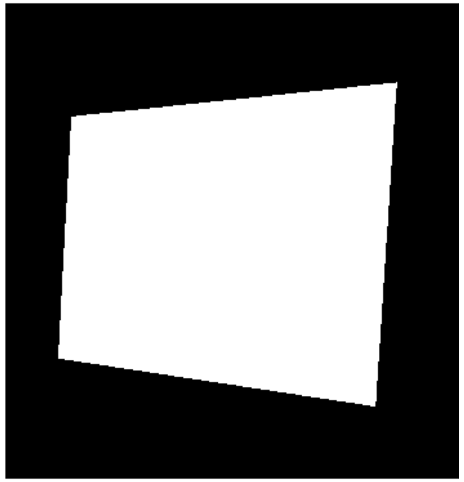
projection result

[TODO 3]

- Hint: You will need to create the array `img_poly` which has the same shape as that of the target image. The shape of `img_poly` can be either (`ht_dst,wid_dst`) or (`ht_dst,wid_dst,3`).
- The pixels within the region formed by (v1,v2,v3,v4) will be set to 1, and the other pixels will be set to zero.
- Use `cv2.fillPoly()` to create `img_poly`.



target image



img_poly

[TODO 3]

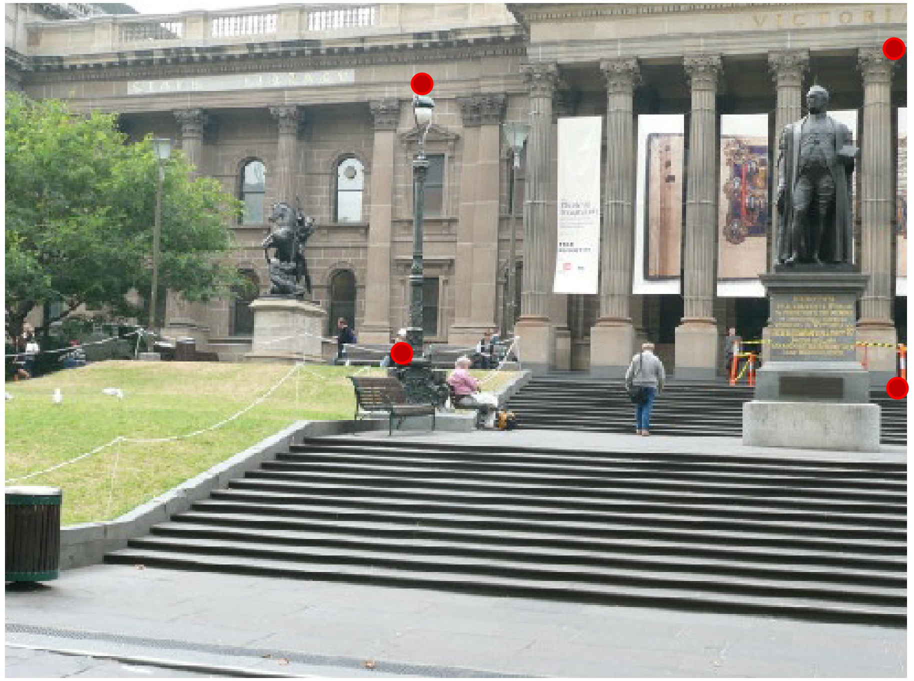
- Let (`ht_src, wid_src`) and (`ht_dst, wid_dst`) be the shapes of source and target images, respectively.

```
G = calc_homography(pts_dst, pts_src)
for j in range(ht_dst):
    for i in range(wid_dst):
        If the pixel of img_poly at $(i, j)$ is greater than 0:
            Use the homography matrix G to project $(i, j)$, and denote the result by $(x, y)$.
            if y < ht_src and x < wid_src:
                Copy the pixel of the source image at $(x, y)$ to the pixel of the target image
                at $(i, j)$.
```

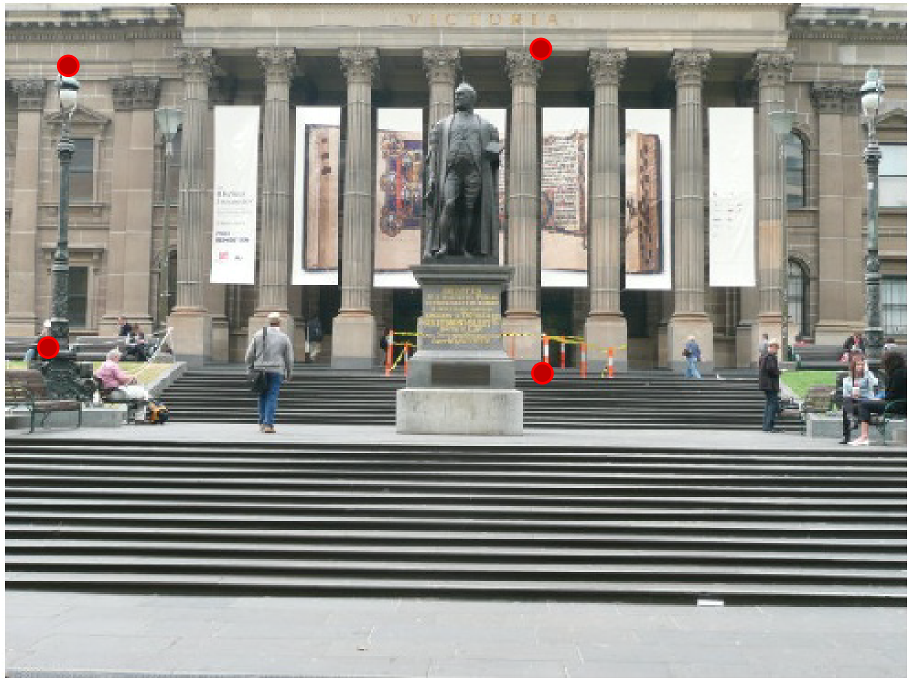
- Remark: The pixel at (i, j) means that the pixel is located at the ith column and jth row of an image.

[TODO 4] Select key points in library1.jpg and library2.jpg.

- Select four points (red points) in library1.jpg, and save the four points in the array `pts_dst`.
- Select four points (red points) in library2.jpg, and save the four points in the array `pts_src`.



library1.jpg



library2.jpg

H

Image credit: Terry Duell

[TODO 5] Warping library2.jpg to create img_warp.

Step 5.1 Compute the homograpy matrix H mapping `pts_dst` to `pts_src`.

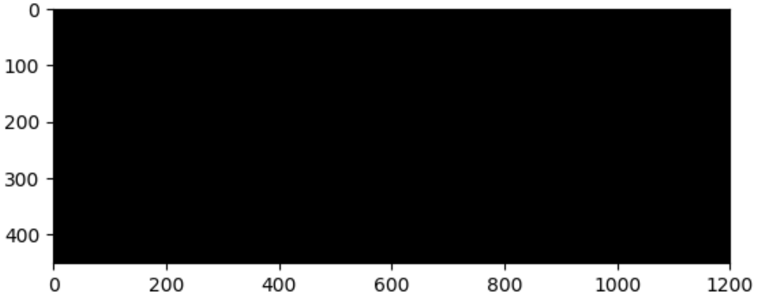
Step 5.2

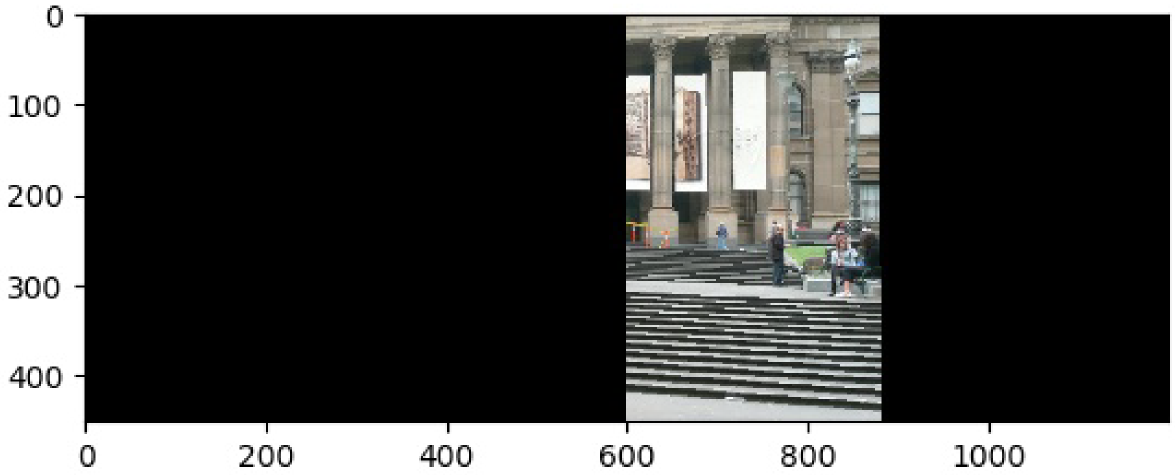
- Let the shapes of library1.jpg and library2.jpg be (`ht_dst, wid_dst`) and (`ht_src, wid_src`).
- Create the zero array `img_canvas` of shape (`ht_dst, wid_dst+wid_src, 3`).

Step 5.3 Similar to the steps in TODO 3, use H to project the whole library2.jpg on `img_canvas[:,wid_dst:,:]`, and the result is denoted by `img_warp` as shown below.

library2.jpg

img_canvas



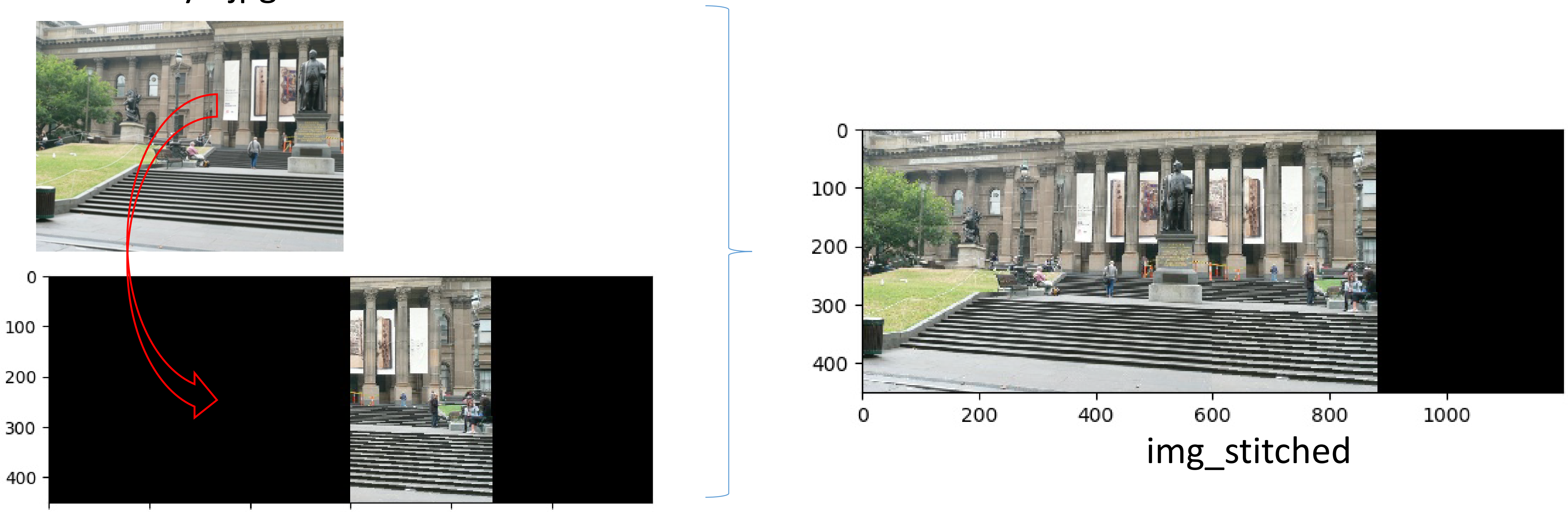


img_warp

[TODO 6] Stitching library1.jpg and img_warped.

- Replace the first few columns of `img_warped` with `library1.jpg` to form the array `img_stitched` as shown in the following figure.

library1.jpg

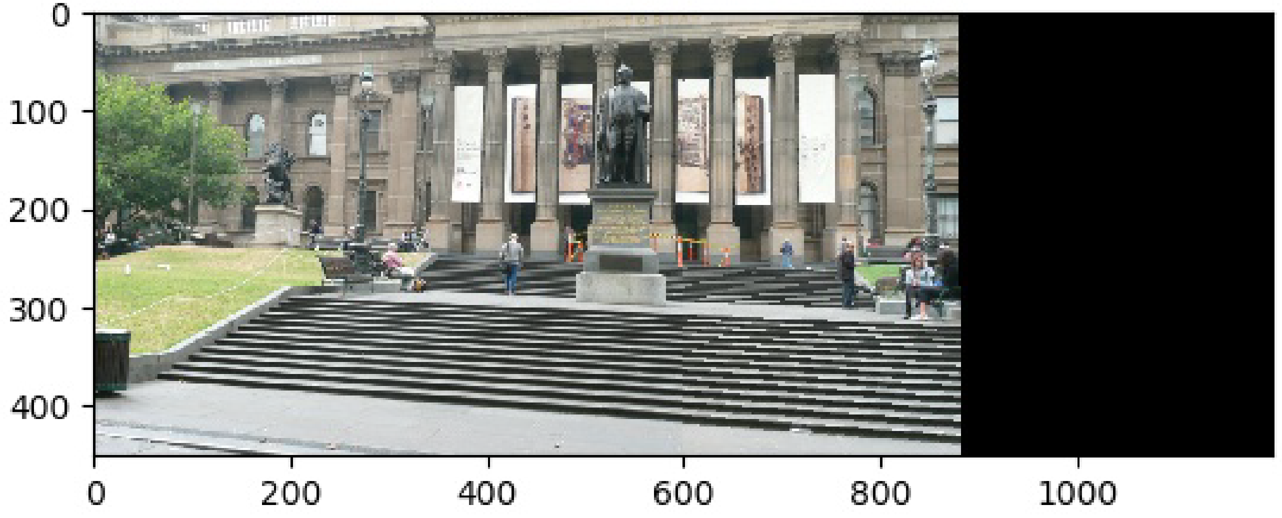


img_stitched

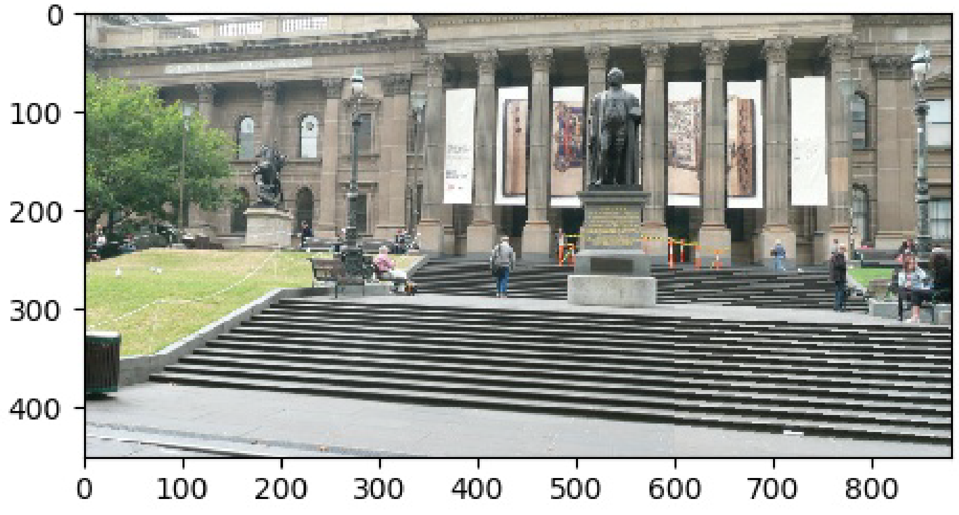
img_warped

[TODO 7] Post-processing the stitched result.

- Create the array `img_stitched_final` containing not all-zero columns of `img_stitched` as shown in the following figure.
- Hint: In Step 5.3, when copying source pixels to target pixels, you need to determine the indices of target pixels. Based on the indices of target pixels, you can reduce `img_stitched` to `img_stitched_final`.



img_stitched



img_stitched_final

[TODO 8] Complete the same task as in TODO 3, but without using any for loops.

- Hint:
- `roi_poly = np.argwhere(img_poly > 0)`

  where `roi_poly` is of shape (1882976, 2)
- Note that `(roi_poly[i,1], roi_poly[i,0])` is the ith point in the target region formed by (v1,v2,v3,v4).
- Create an array `roi_dst` of shape (2, 1882976) such that `(roi_dst[0,i], roi_dst[1,i])` is the ith point in the target region formed by (v1,v2,v3,v4).
- Use the homography matrix H to convert each column of `roi_dst`,

  and denote the converted result as `roi_src`, where `roi_src` is of shape (2, 1882976)
- Create the Boolean array `idx` of shape (1882976,) such that

  all entries of `roi_src[0,idx]` are less than `wid_src`, and

  all entries of `roi_src[1,idx]` are less than `ht_src`
- Copy source pixels to target pixels based on `roi_dst`, `roi_src`, and `idx`.