Panoramic Stitching with Homographies

Syed Zaidi

April 2023

1 Intro to homographies

We have three different transformations to apply on the images:

1. Rotation (clockwise, 10 degrees):

$$M_{rotate} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   where $\theta = -10°$ (negative angle for clockwise rotation).

2. Translation (100 pixels right):

$$M_{translate} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

   where $t_x = 100$ and $t_y = 0$.

3. Scaling (shrink by half):

$$M_{scale} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

   where $s_x = 0.5$ and $s_y = 0.5$.

We apply these transformations using the function `cv2.warpPerspective()`.

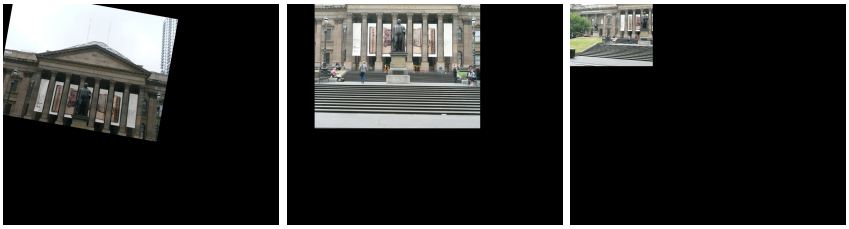

Figure 1: The figure the show in order the image1 rotated, the image2 translated, and the image 3 scaled

2 Panoramic Stitching

In this section, we present a panoramic stitching algorithm that combines multiple images into a seamless, wide-angle view. The algorithm is divided into four main steps: computing SIFT features, matching features, estimating homographies, and warping and translating images.

2.1 Compute SIFT Features

The Scale-Invariant Feature Transform (SIFT) is used to detect and describe local features in images. In this step, we compute the SIFT features for each of the input images, which will later be used to find corresponding features in different images.

```
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)
kp3, des3 = sift.detectAndCompute(img3, None)
```

2.2 Match Features

In this step, we match SIFT features in image 2.jpg with corresponding features in images 1.jpg and 3.jpg. The FLANN-based matcher is used to find the most similar features in terms of $l2$ distance. After finding the matches, we apply Lowe's ratio test to filter out poor matches and retain only the 100 best matches.

```
best_matches_12 = find_best_matches(des1, des2)
best_matches_23 = find_best_matches(des2, des3)
```

2.3 Estimate Homographies

With the matched features from the previous step, we estimate the homographies using the RANSAC algorithm. The homographies map the matched features in image 1.jpg to those in image 2.jpg, and those in image 3.jpg to those in image 2.jpg. The reprojection error threshold for RANSAC is set to 2 pixels.

```
H12, _ = cv2.findHomography(src_pts_12, dst_pts_12, cv2.RANSAC, 2)
H32, _ = cv2.findHomography(src_pts_32, dst_pts_32, cv2.RANSAC, 2)
```

2.4 Warp and Translate Images

In this final step, we use the homography matrices calculated in the previous step to align the images. Image 1.jpg is aligned with image 2.jpg, and image 3.jpg is aligned with image 2.jpg. All images are then translated 350 pixels to the right and 300 pixels down. The aligned and translated images are fused using the np.maximum function to create a seamless panoramic view.

```
img1_warped = warp_and_translate(img1, H12, translation)
img2_warped = warp_and_translate(img2, np.eye(3), translation)
img3_warped = warp_and_translate(img3, H32, translation)
fused_image = np.maximum(img1_warped, np.maximum(img2_warped, img3_warped))
```

The resulting panoramic image is a seamless, wide-angle view created by combining the three input images. Some minor ghosting artifacts may be present, but the details around the edges of the images should align quite well.

Figure 2: The figure show the fused image according to the requirements