Image Stitching

Davide Laureti
Department of Computer Science, Keio University

June 9, 2024

Abstract

This report describes the process and outcomes of stitching images together. The objective was to create an image stitching with multiple images using computer vision techniques.

1 Introduction

Goal: The goal of this project is to create an image stitching from multiple images. An Image stitching consists of combining multiple images into one combined canvas, where each images is aligned with each other such that the image appear seamless and coherent. It is commonly used for panoramic photos, when we take multiple picture of a panorama and it's stitched into one single wide photo; or it's used for photo sphere, where you can take pictures at each orientation and they are stitched together to create an immersive 3D sphere of the surroundings.

2 Method

2.1 Equipment

I used my personal smartphone a Google Pixel 6a. I used his main camera.

2.2 Key point detection

The first step is key point detection, where an algorithm scan the picture to look for keypoint, points that are interesting like a corner of a table. Key point detection is run on both the images. There are multiple key point detection algorithms like SIFT, ORB etc. They both find keypoints, the main difference is the encoding of the descriptor of the keypoints: SIFT encodes it into floating point values, for which distances is computed ad Euclidean distance, while ORB into binary strings, for which distance is computed as Hamming distance.

2.3 Feature matching

After detecting points, we need to match features. Matching features mean to find the corresponding keypoints of the first image keypoints. Usually we are not able to match every keypoint, maybe because the algorithm didnt find it or maybe because that part of the first image is not even present in the second image. We can use different strategies for Feature Matching, for example Brute force. Brute force matching consist of calculating the distance between each pair of keypoint, one from the first image, one from the second image. Then each keypoint get matched with the keypoint with which they have the lowest distance. This algorithm guarantee to find the best answer but it's compuationally expensive, it's a $O(n^2)$, where n is the number of keypoints. This is feasible for small use cases but for numerous images, we use approximate algorithms, the most famous one is FLANN. FLANN in openCV support the use of 6 different algorithm, for example the one based on KD tree, suited for floating points descriptors, and LHS hashing, suited for binary desciptors. We will use the second one. There are cases where Another strategy is to run a K-Nearest Neighbors search, with k = 2, and apply Lowe's filter on the result. The 2-NN search for each matched keypoint returns the 2 closest matches. Then we apply Lowe's filter, which compare the two distances between the found matches for each keypoint. If the difference of distances is greater than a certain ratio, it means that the best match is robust enough, while if it's lower than the ratio, it may be a false positive, so we discard this wrong match.

2.4 Transform estimation

After having found keypoint macthes we need to choose a model of transformation and calculate the parameters. We analyze 4 different model: translation, similarity, affine and homography. After computing the matrices we warp them into the canavs, calculate the combined canvas size and display the stitched image.

3 Results

Two images were captured and uploaded into the script. The keypoints and the feature matching was performed successfully. The matrix estimation result were plausible give as a result the correct stitched image, with increasing more accuracy with more parameters into the model. The images showing the results and parameters values can be found at the end of the document.

4 Discussion

The images in the results section shows how using different models changes the final result of the image stitching. We notice how with the translation model we account only for translation, therefore the images shows only an horizontal movement. With Similarity we account also for rotations and scaling, and in fact we successfully see a small rotation. With the Affine model we include also the shearing effect, some displacement that transform a rectangle image into a parallelogram. In our case, the difference was not noticeable, by looking at the matrices we see that the difference in translation is only of 5 pixels. Using Homography we take in consideration every kind of transformation, including projective transformations. We in fact how the image is transformed, having the right side enlarged compared to the left side.

We also compare the stitching we performed with a state of art technique. We take as example the Autostitch from University of British Columbia. Their stitching is somehow transformed to be seen in a better way, while the one we performed fix the first image as a reference and then add the second image warp.

5 Conclusion

We created an image stitching, performing successfully every steps of the procedure. We couldn't appreciate the difference between similarity model and affine model but this is regarded as an input characteristic. We finally compared our result with a state of art software.

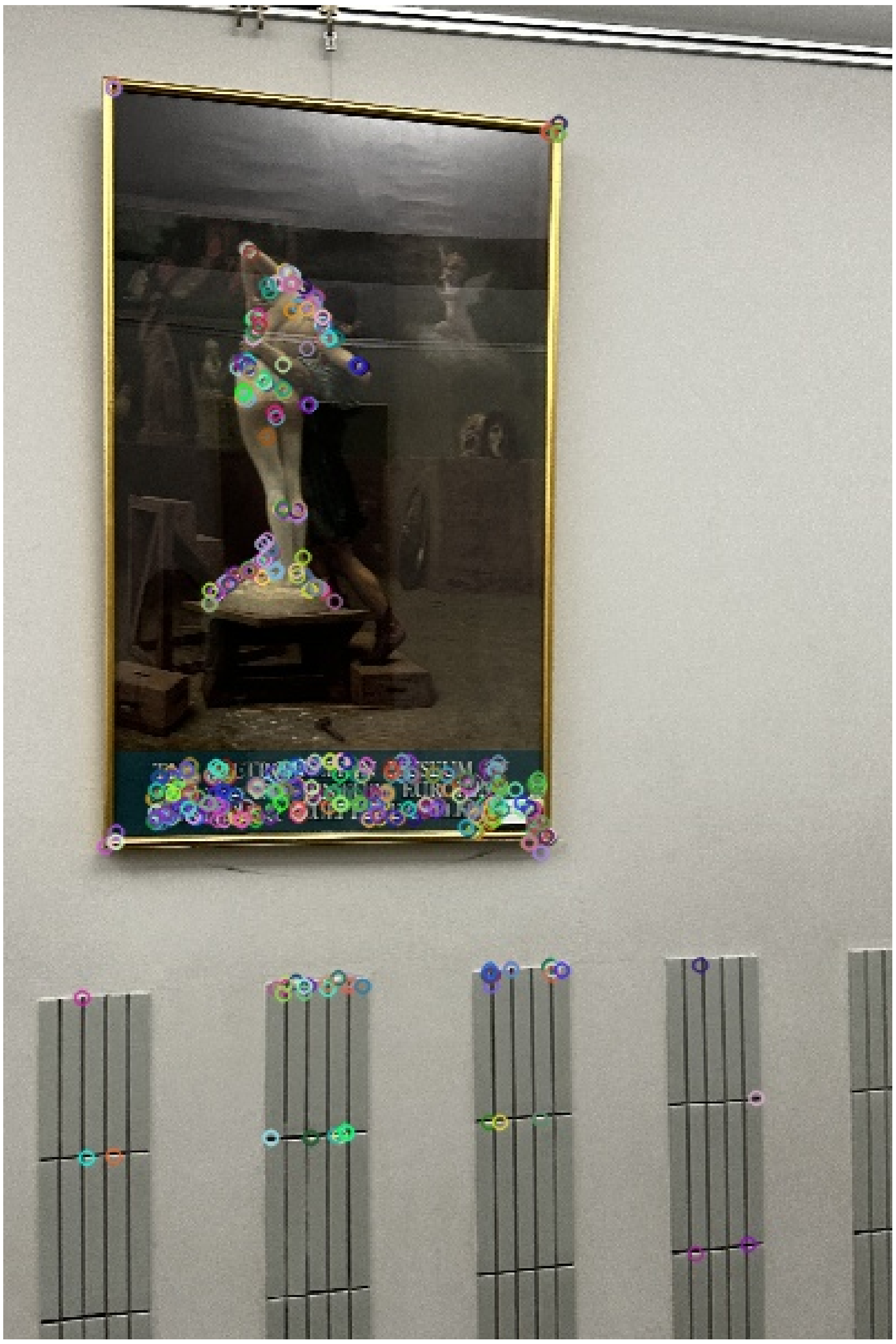


Figure 1: First image with keypoints drawn

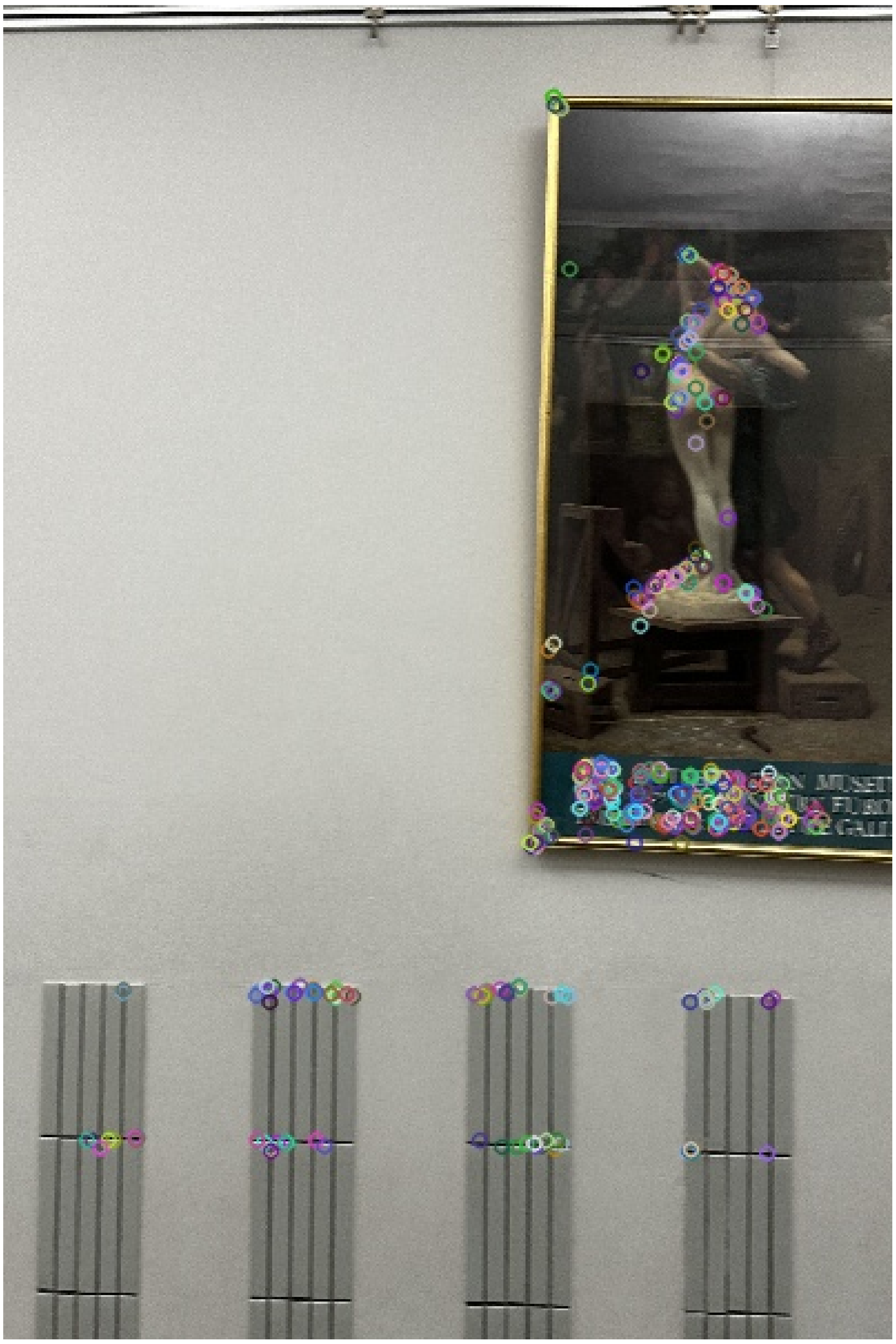


Figure 2: Second image with keypoints drawn

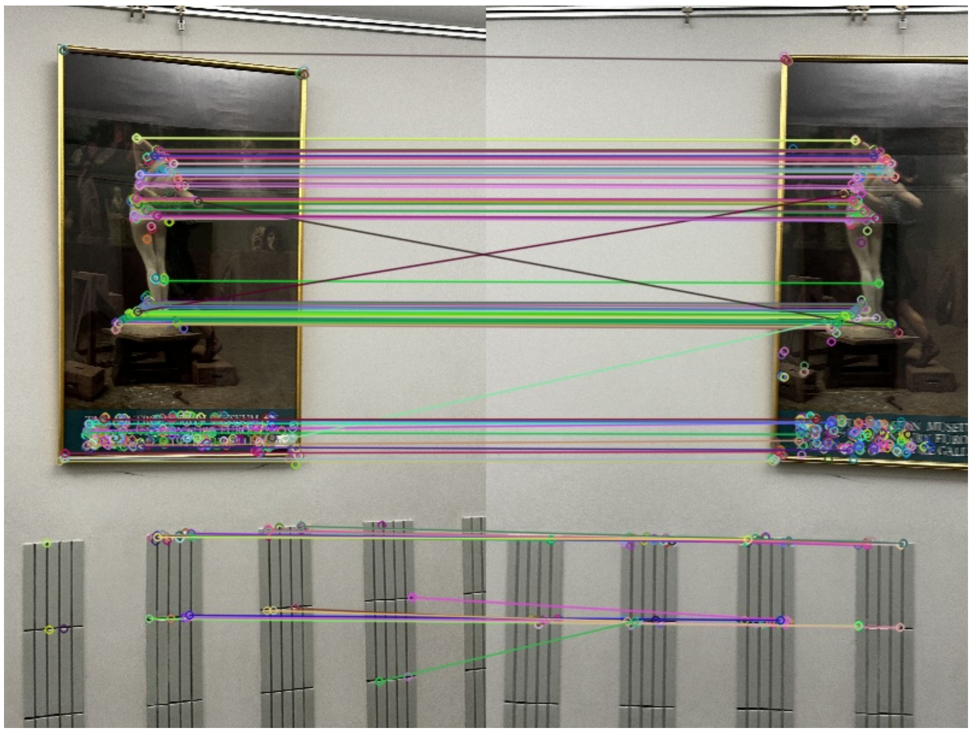
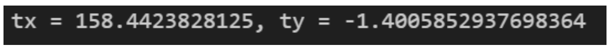
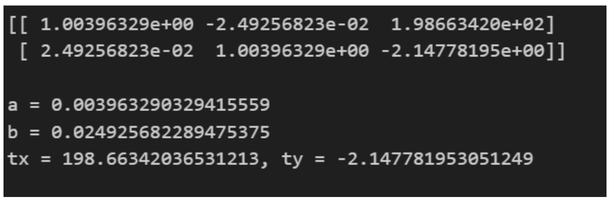


Figure 3: Feature matching, using Brute Force approach and drawing the best 100 matches



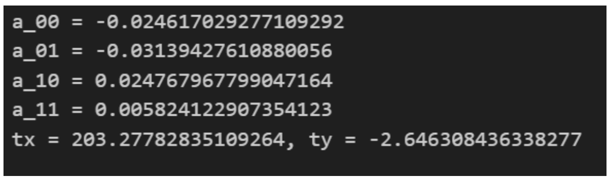
```
tx = 158.4423828125, ty = -1.4005852937698364
```

Figure 4: translation parameters result



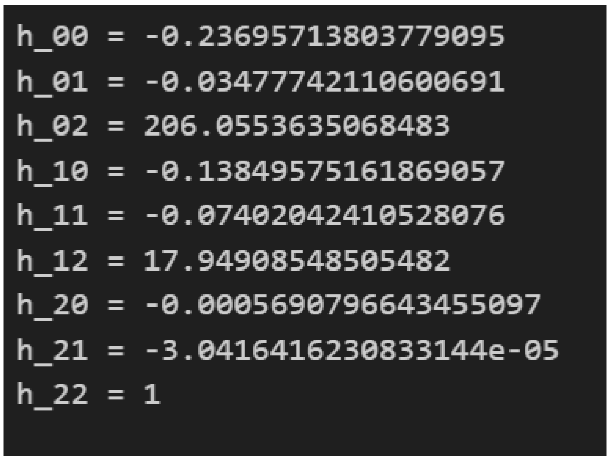
```
[[ 1.00396329e+00 -2.49256823e-02  1.98663420e+02]
 [ 2.49256823e-02  1.00396329e+00 -2.14778195e+00]]

a = 0.003963290329415559
b = 0.024925682289475375
tx = 198.66342036531213, ty = -2.147781953051249
```

Figure 5: Similarity matrix result



```
a_00 = -0.024617029277109292
a_01 = -0.03139427610880056
a_10 = 0.024767967799047164
a_11 = 0.005824122907354123
tx = 203.27782835109264, ty = -2.646308436338277
```

Figure 6: Affine matrix result



```
h_00 = -0.23695713803779095
h_01 = -0.03477742110600691
h_02 = 206.0553635068483
h_10 = -0.13849575161869057
h_11 = -0.07402042410528076
h_12 = 17.94908548505482
h_20 = -0.0005690796643455097
h_21 = -3.0416416230833144e-05
h_22 = 1
```

Figure 7: Homography matrix result

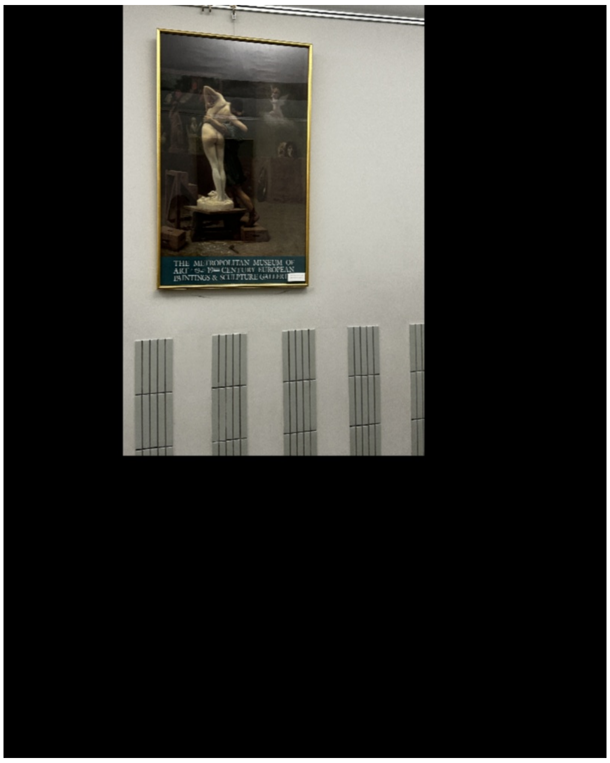


Figure 8: Translation image result

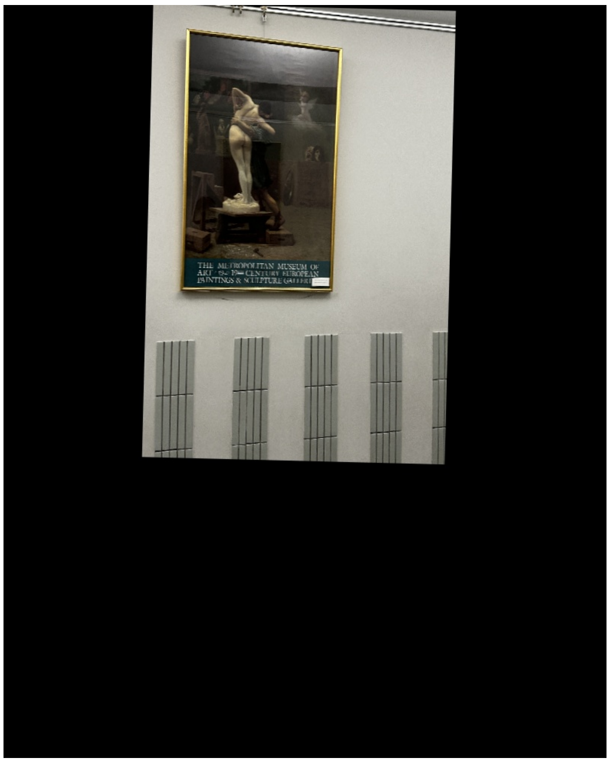

THE METROPOLITAN MUSEUM OF
ART 19th CENTURY EUROPEAN
PAINTINGS & SCULPTURE GALLERIES

Figure 9: Similarity image result

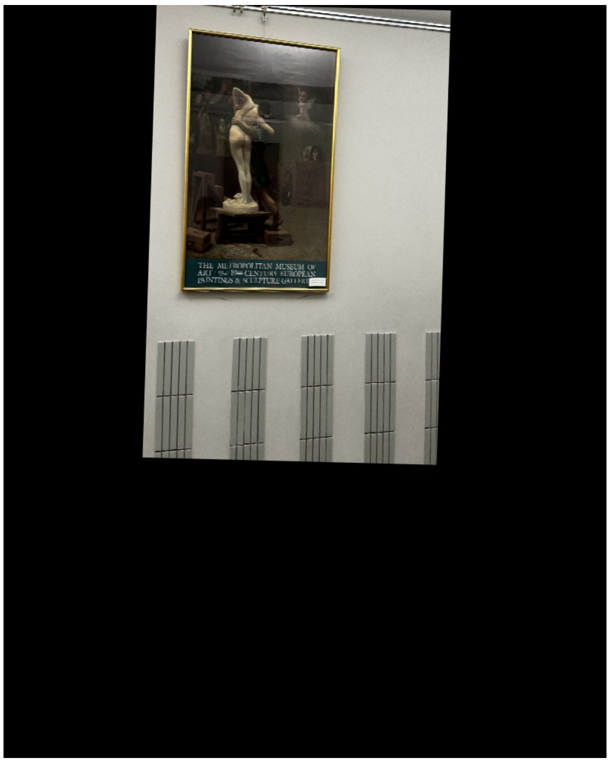

THE METROPOLITAN MUSEUM OF
ART 19th CENTURY EUROPEAN
PAINTINGS & SCULPTURE GALLERIES

Figure 10: Affine image result

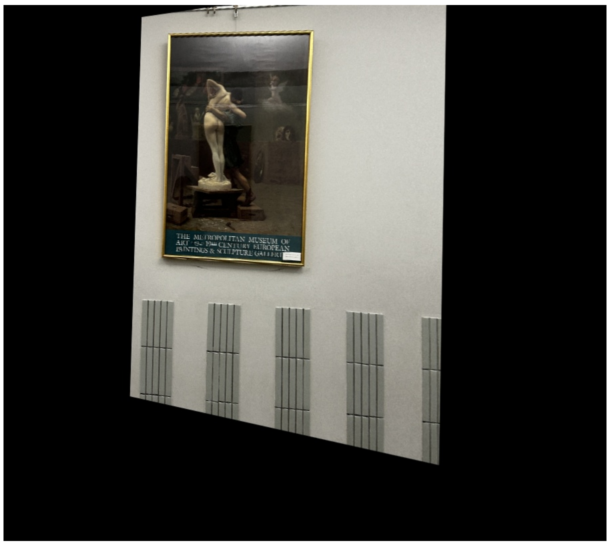

THE METROPOLITAN MUSEUM OF
ART · 19th CENTURY EUROPEAN
PAINTINGS & SCULPTURE GALLERI

Figure 11: Homography Image result

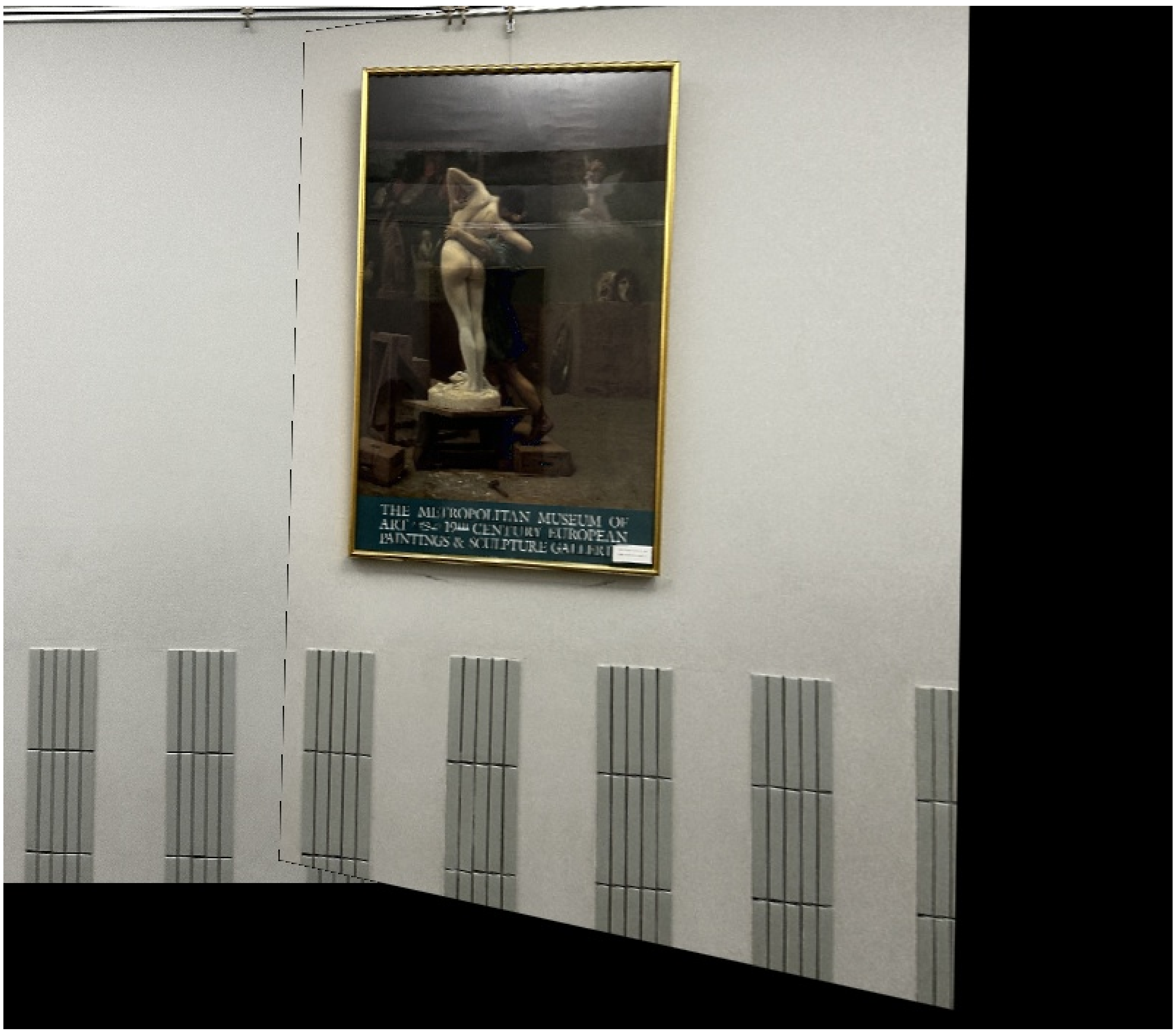

THE METROPOLITAN MUSEUM OF
ART · 19th CENTURY EUROPEAN
PAINTINGS & SCULPTURE GALLERI

Figure 12: Final stitching using Homography model

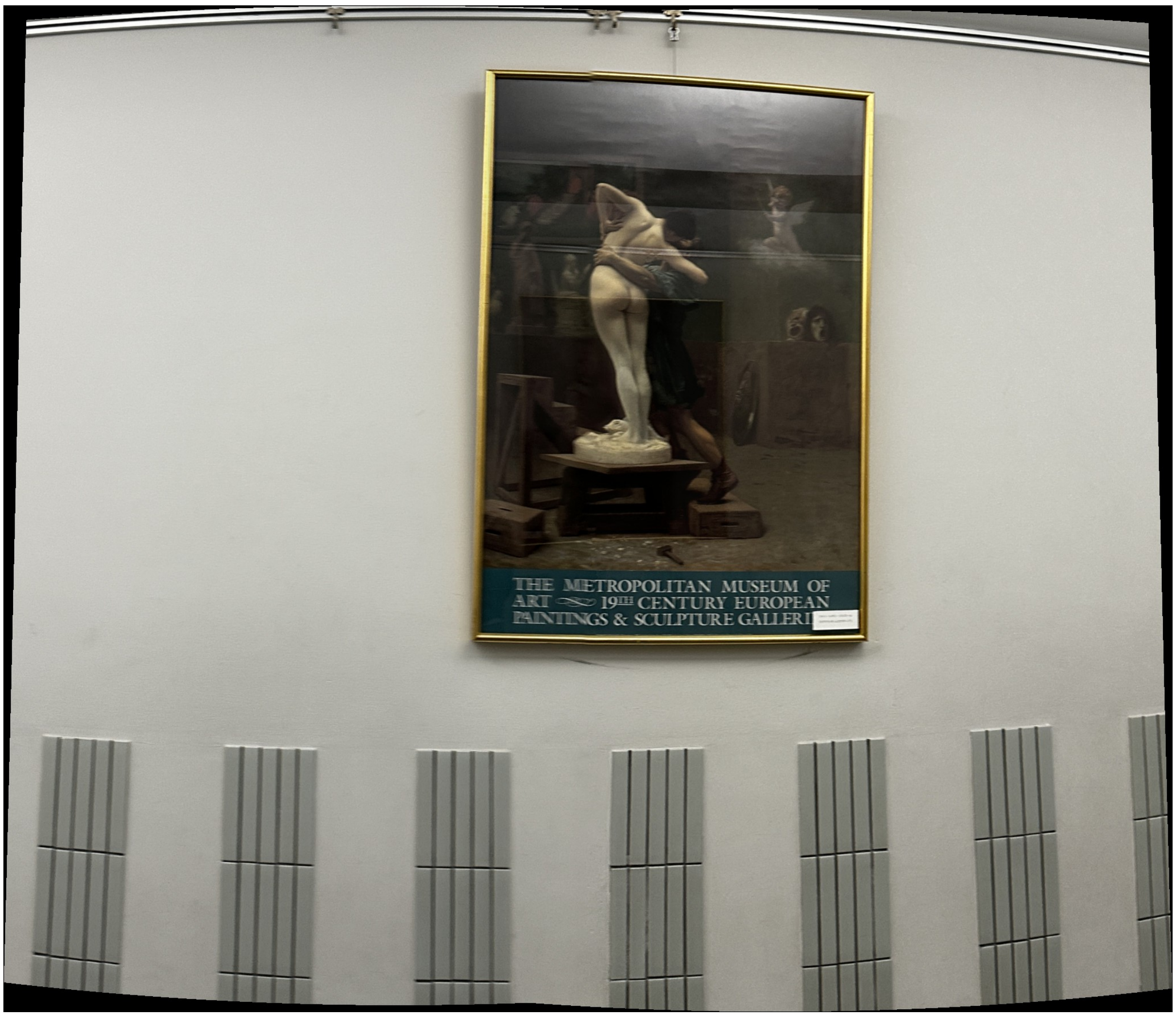

THE METROPOLITAN MUSEUM OF
ART 19TH CENTURY EUROPEAN
PAINTINGS & SCULPTURE GALLERI

Figure 13: AutoStitch image stitching