Gal Oren 206232506
Ohad Yaari 305337545
January 3, 2021

TECHNION
Israel Institute
of Technology

046746 – Computer Vision

Homework Assignment: 2
Classifiers and Segmentation

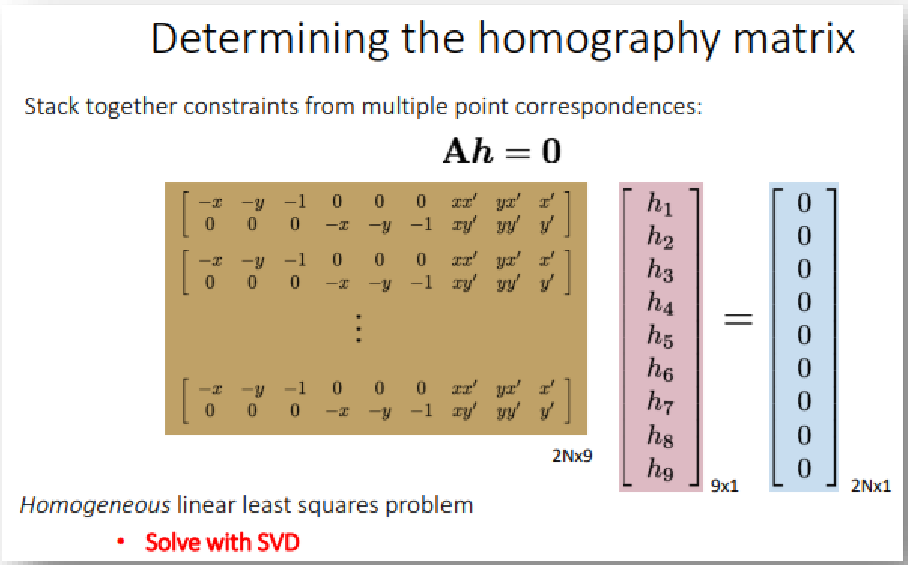
Part 1 – Planar Homographies: Practice

Section 1 – Manually finding corresponding points

for picking the points we used color images instead of grayscale

Section 2 – Calculate transformation

We followed the method we saw in the lecture for finding H:



Determining the homography matrix

Stack together constraints from multiple point correspondences:

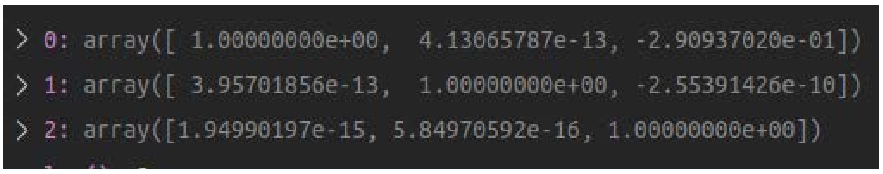
$$\mathbf{A}\boldsymbol{h} = \mathbf{0}$$

$$\begin{bmatrix} \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \\ \vdots \\ \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix} \end{bmatrix}_{2N\text{x}9} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}_{9\text{x}1} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}_{2N\text{x}1}$$

Homogeneous linear least squares problem

- Solve with SVD

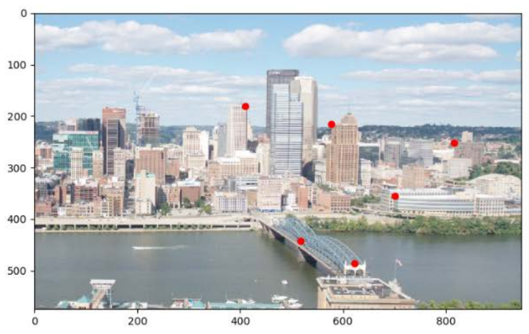
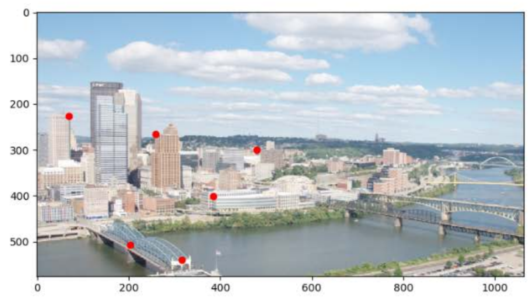
That means we used the corresponding points we found in section 1 and got A matrix. Next, we did SVD on A matrix and took the eigenvector corresponding to the smallest eigenvalue.
For sanity check we tried to find the transformation matrix between image to itself and we got after normalization with the bottom right value:



```
> 0: array([ 1.00000000e+00,  4.13065787e-13, -2.90937020e-01])
> 1: array([ 3.95701856e-13,  1.00000000e+00, -2.55391426e-10])
> 2: array([1.94990197e-15, 5.84970592e-16, 1.00000000e+00])
```

As expected, we got a matrix that close to identity (the upper right value is a little high due to the error in the manual selection of the corresponding points).

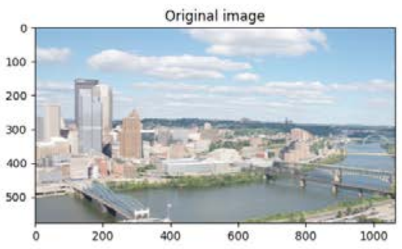
Moreover, we were asked to show the projection of arbitrary points between the incline images.
The results:

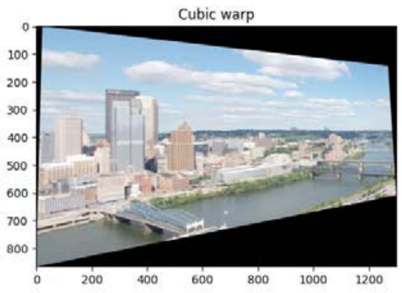



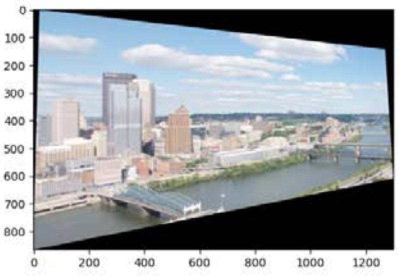
As we can see, the projection accuracy is good.

Section 3 – Image warping

OpenCV's warpPerspective function is implementing inverse mapping in order to avoid sampling artifacts such as mapping a pixel to a irrational value.
We warp incline_R image in both linear and cubic interpolation methods and we got:







The linear warp preforms bilinear interpolation which means:

Where f is the new image and g_s is the original image.

The cubic warp preforms bicubic interpolation which means applying a convolution of each dimension with the function:

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{for } |x| \leq 1, \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{for } 1 < |x| < 2, \\ 0 & \text{otherwise,} \end{cases}$$

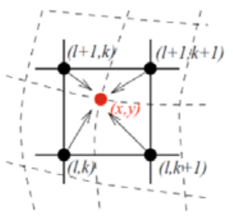
Where a is usually set to -0.5 or -0.75 (from Wikipedia)

The cubic interpolation produce a sharper image than the linear interpolation but its cost more computation time.

Bilinear interpolation

$$\begin{aligned} f(x,y) &= (1-a)(1-b)\, g_s(l,k) + a\,(1-b)\, g_s(l+1,k) \\ &+ b\,(1-a)\, g_s(l,k+1) + ab\, g_s(l+1,k+1)\,, \end{aligned}$$

$$l = \text{ceil}\,(x)\,, \quad a = x - l\,,$$
$$k = \text{ceil}\,(y)\,, \quad b = y - k\,.$$

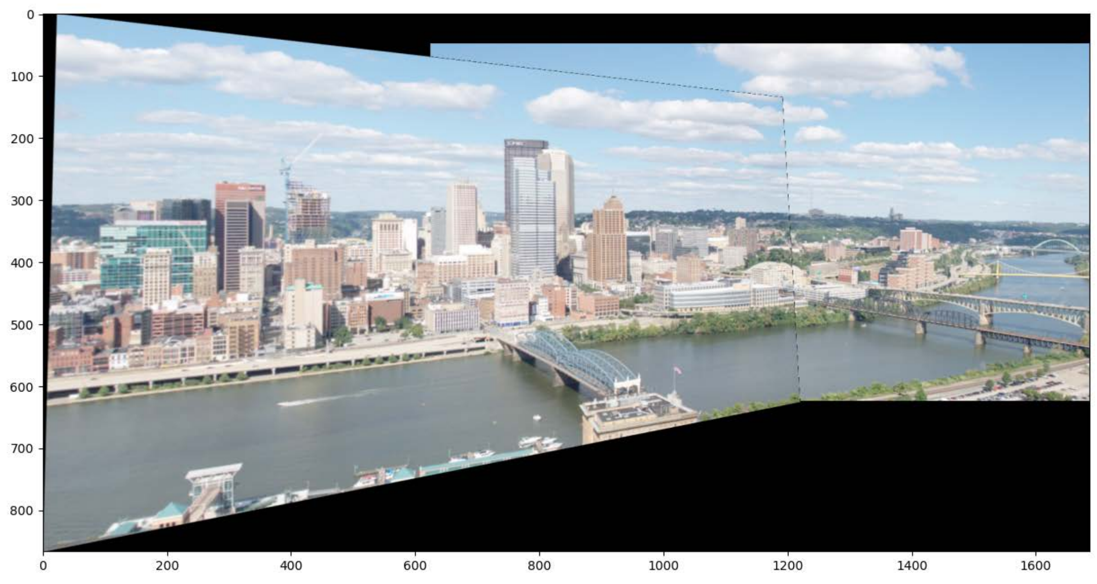


(Taken from: https://stackoverflow.com/questions/50086717/bilinear-interpolation-artifacts)

In order to ensure that the entire image is being shown after the warp (and it isn't being cut)
We first look where the points of the border of the image is mapped (To be more precise, we find the points at the edges of the image where the pixels are different from 0 - because after making a stich between the images the resulting image contains blank areas). Finding the map points done by the **cv2.perspectiveTransform** function which return the mapped points according to the homography matrix . than we translate images 1 and 2 according to the max and min of the borders of the mapped source image points and the destination image so they will be in the same perspective and at last apply the prespective transform to the source image and set its size according to max and mins points, we apply this operation in the wrapH and wrapH2Images functions .

Section 4 – Panorama stitching

We stitched the images by putting it one above another and got the following panorama:



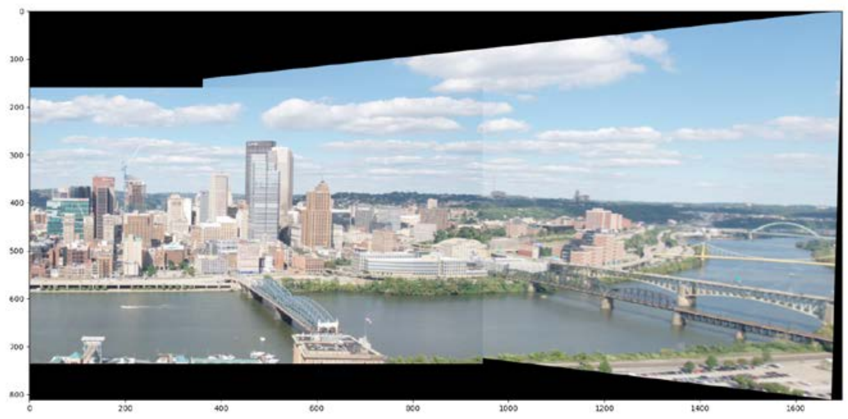
1.5 - Autonomous panorama stitching using SIFT

In this part instead to create the panorama from holography matrix calculated based on manually selected points we used the Sift descriptors to find those points autonomously.
To match the resulting points and get correspondences we used Brute-Force matcher with knn-2, the principle of the BF-matcher is simple It takes the descriptor of one feature in first set and is matched with all other features in second set using 2-norm distance and return the two closest descriptors which it find.

We taked k=2 so that we can apply ratio test explained by D.Lowe in his paper.
in simplicity that law assume that between two images there is one to one matching relation between descriptors and there isn't descriptor in one image that will fit to two descriptors in the second image so there needs to be enough difference between the best and second-best matches.
the threshold for rejection set by calculate the ratio of the distances of the two descriptors.
D.lowe suggest in his paper value of 0.75 but we found it too high and set the default value to 0.3.

After finding the matched point we did the same process as before- calculate the H matrix according the matched points ,warp the source image against the destination image ,and stich both images together.

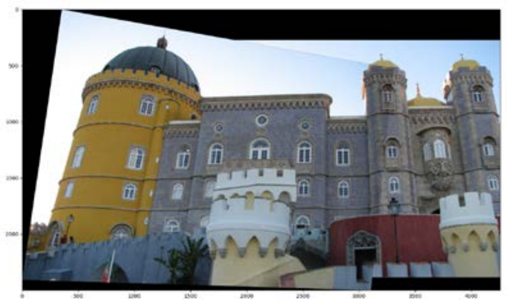
We got the following panorama:

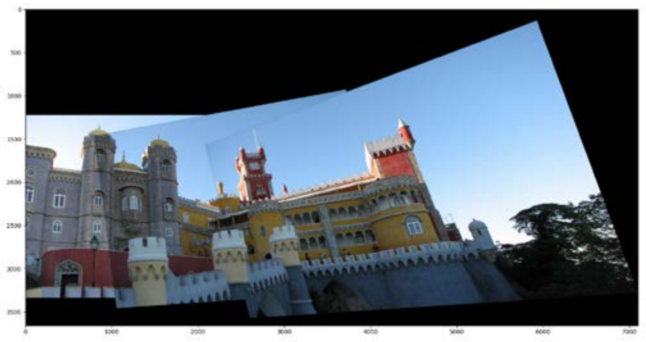


1.6 - Compare SIFT and Manual image selection:

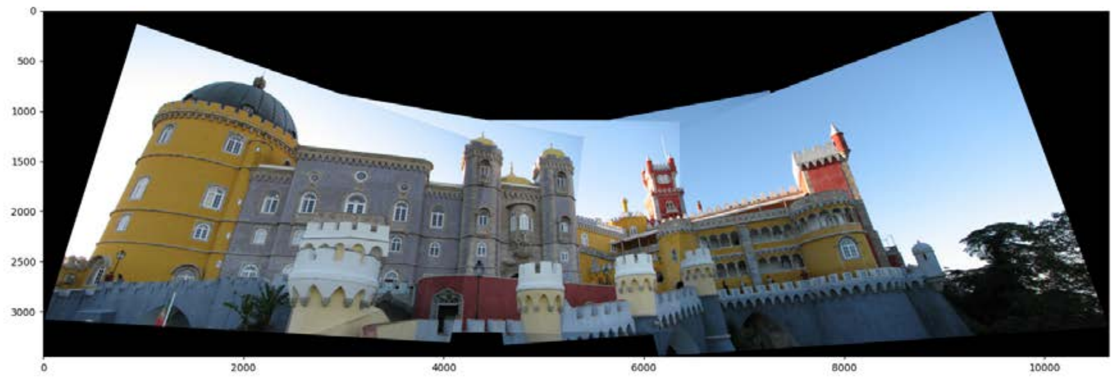
In this method we compared the performance of the sift algorithm with the manual by creating two panoramas consisting of 5 images - the first of a beach and the second of Pena National Palace.

The process of creating the complete panorama consisting of 5 images is divided into the creation of a panorama for the right and left halves and finally the projection of the left panorama over the right, from the idea of creating the perspective of the image around the middle image and not around one side.
For example For the Sift Method with Sinatra palace we get the following right and left panoramas :

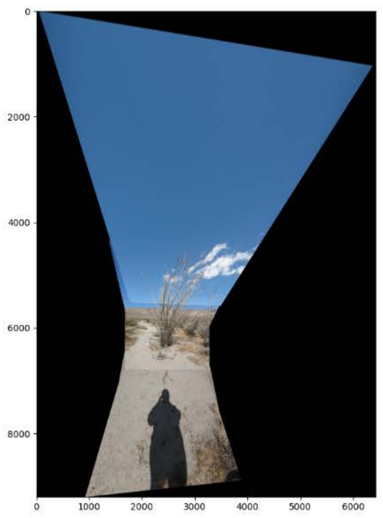




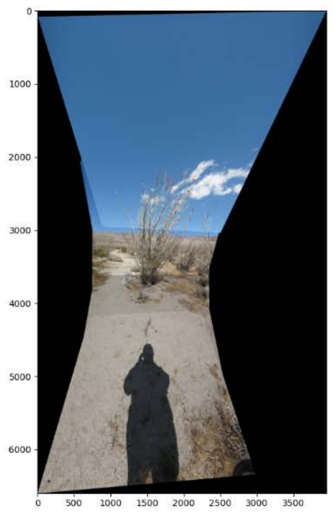
And the final result we get for the sift(with trehshold of 0.15) is :

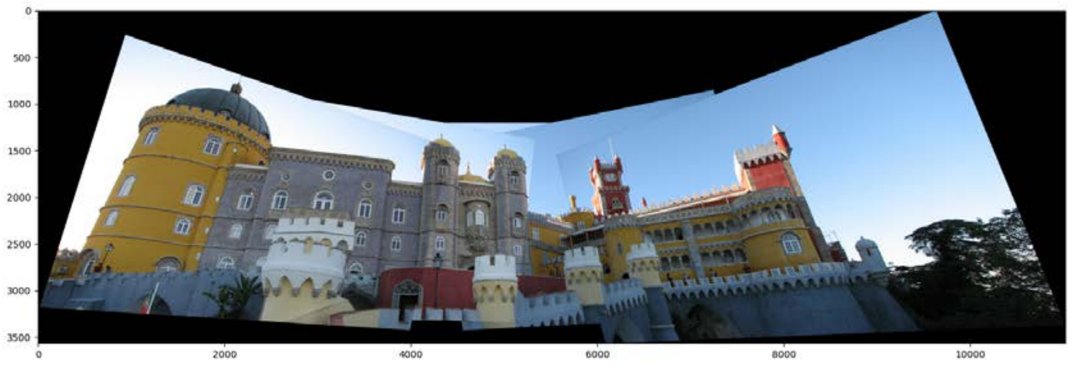


in simillar manner the result for the beach image from the SIFT is :

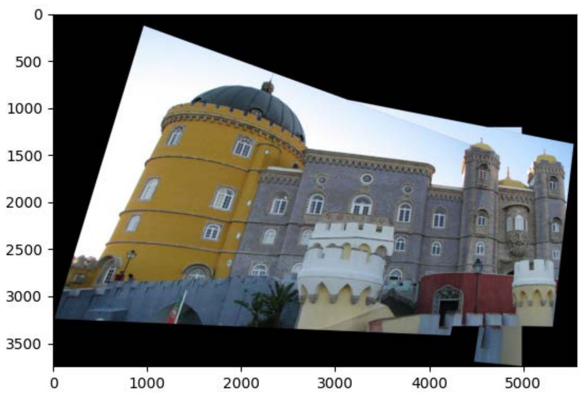


The result we get from the manual selection are the following:

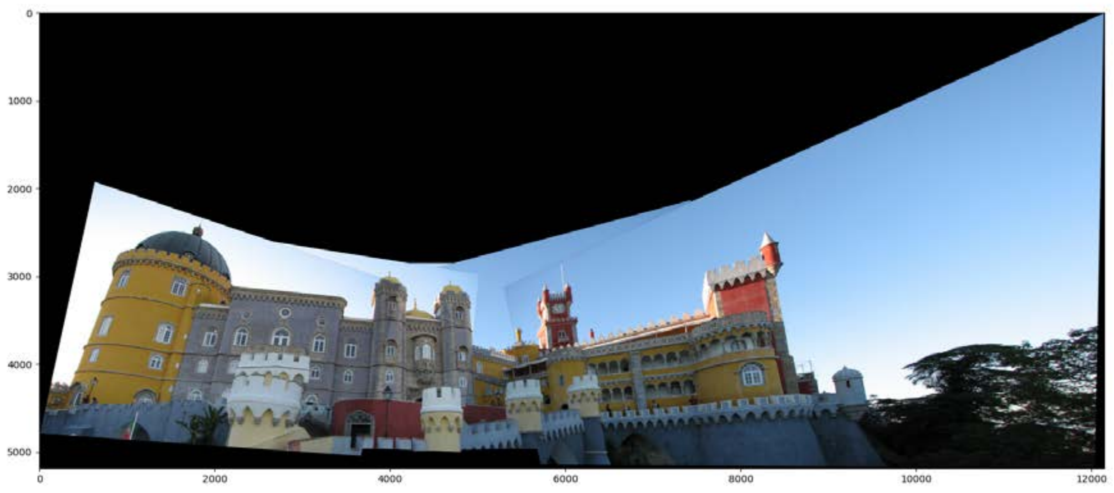




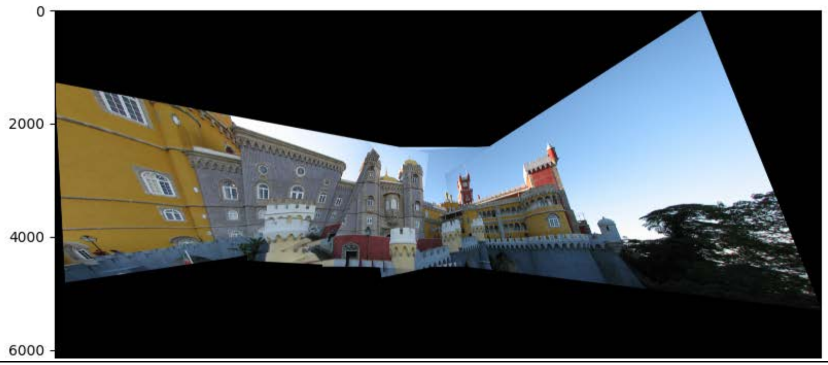
We can see that we get a similar result for the manual and the sift descriptor ,
but that can be explained because the matching method that we do in the sift algorithm ,and accurate selected points in the manual method. We chose the threshold for the D.Lowe to 0.15 which is low value regarding the auteur suggestion for 0.75 , in this way we discard a lot of outliers points from the calculation , we test also for 0.3 and get the following result: as we can see the Sift failed this time .



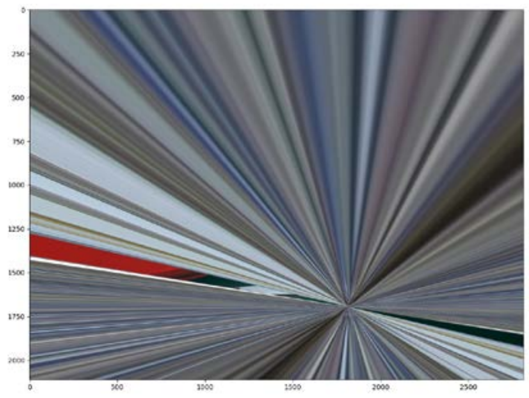
We can see that in both methods the change of perspective between the images is successful and there is a good match but on the other hand we see that there are color differences between image because of different exposure to light and that the boundaries between the images are very noticeable.
The advantage of sift regard the manual way is of course avoiding the need to select the points manually ,but the disadvantage is that the threshold need to take carefully as we can see a little change of this parameter ruined the panorama.
We also test the Sift method without the condition of the distance as before but return only the K-most closet matches and get The following results: k=7



K=4



Randomly selected :



1.7 - RANSAC:

In this part we try to improve the result of the sift and manual selection methods by RANSAC algorithm . the problem with previous methods is that the matching points that returned are not necessarily accurate enough. ransac came up with a solution to this by trying to find the collection of points that would give the best result, this set will be set to one that maximize the number of inliers that were mapped correctly and met the threshold conditions of the distance between the points in the images.

The algorithm takes the following steps:

Loop:

1. Randomly select a group of points
2. Fit a model to the selected group
3. Find the inliers of the computed model
4. If number of inliers is large enough re-compute model using only inliers
5. Compute number of inliers of updated model

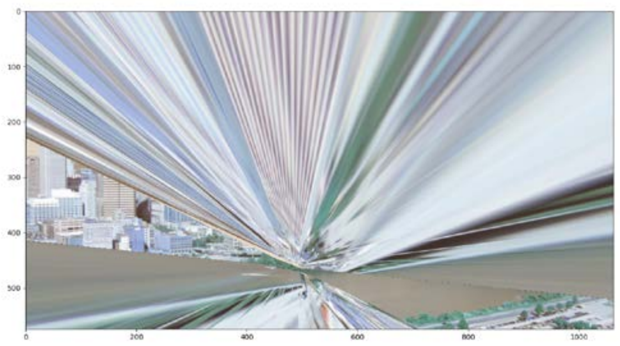
The winner: model with the largest number of inliers

In our implementation we used paramteters of tool=1 anf nItears=1000

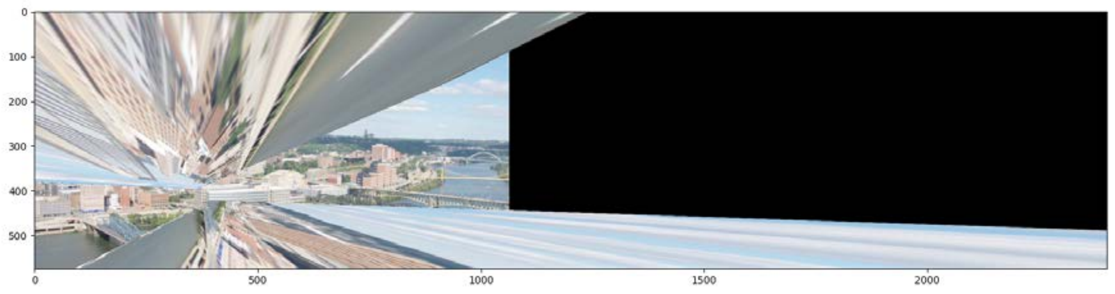
In order to show the necessity of ransac we randomly selected a group of points and compared the results of the two methods with and without ransac.

We test it on incline_L and incline_r images and get the following Results:

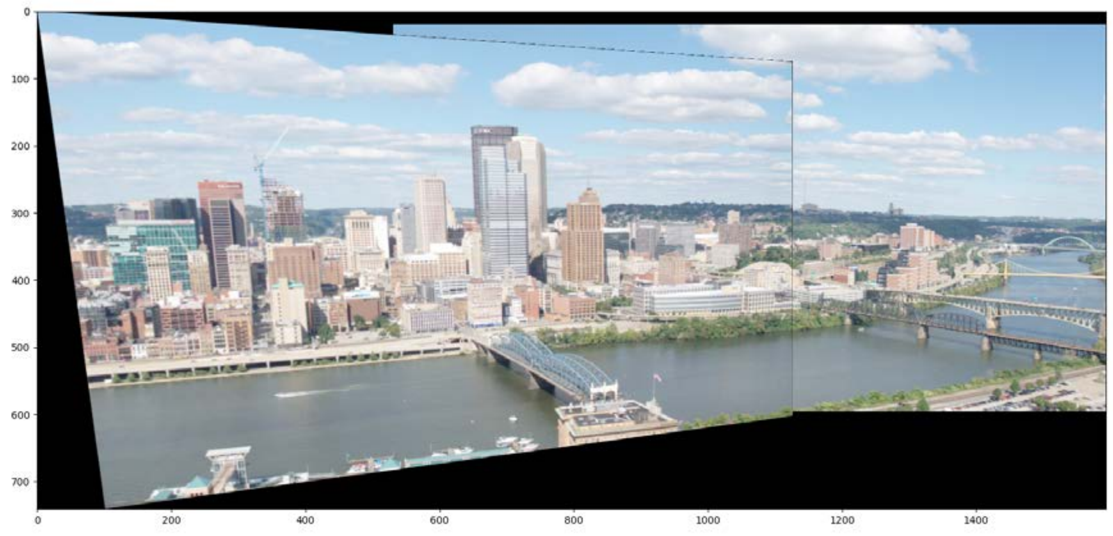
Without Ransac Sift:



Without RANSAC Manual:



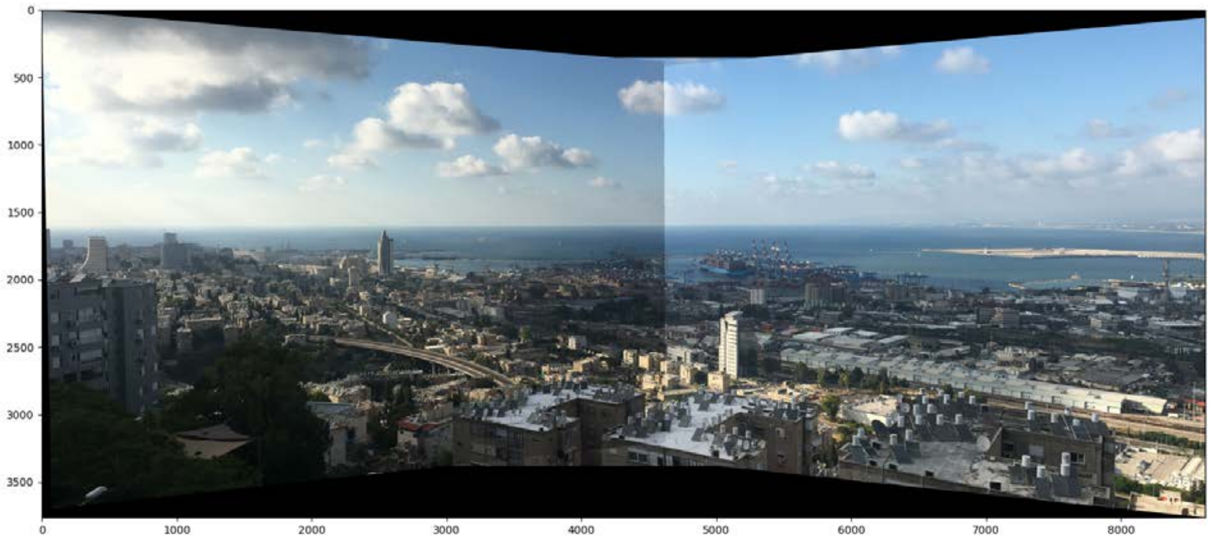
With RANSAC Manual:



We see that ransac caused a significant improvement in the results of the panorama so it can be incorporated as an additional step in the process before calculating the homography matrix, by running it on the matched points obtained.

1.8 - Be Creative:

Panorama of Haifa City.

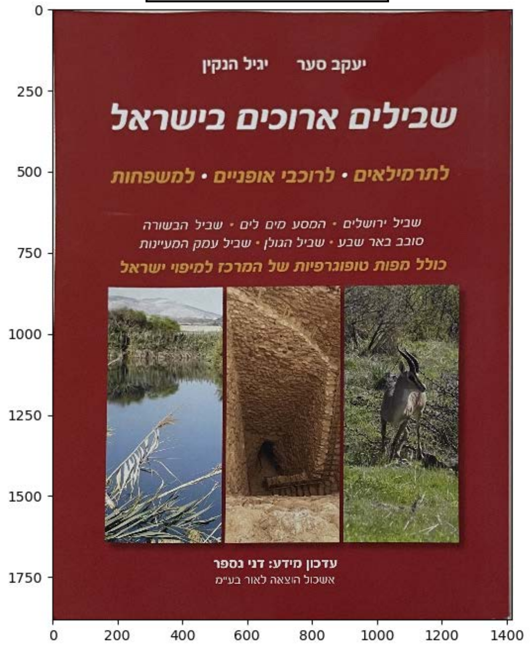


<u>Part 2 – Creating your augmented reality application</u>

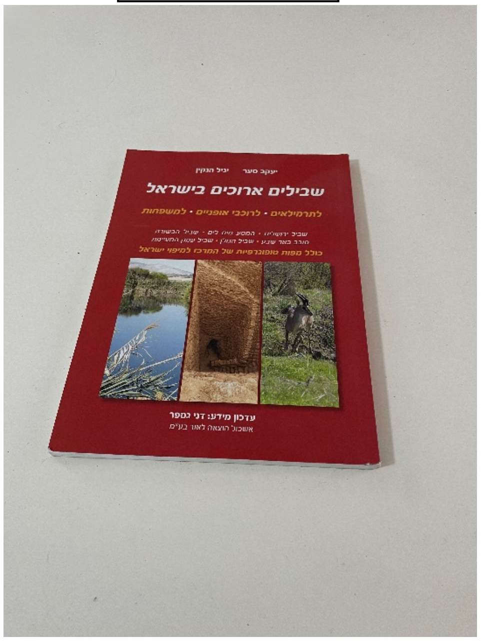
<u>Section 1 – create reference model</u>

Here is the photo we took and the reference image from it:

Reference image

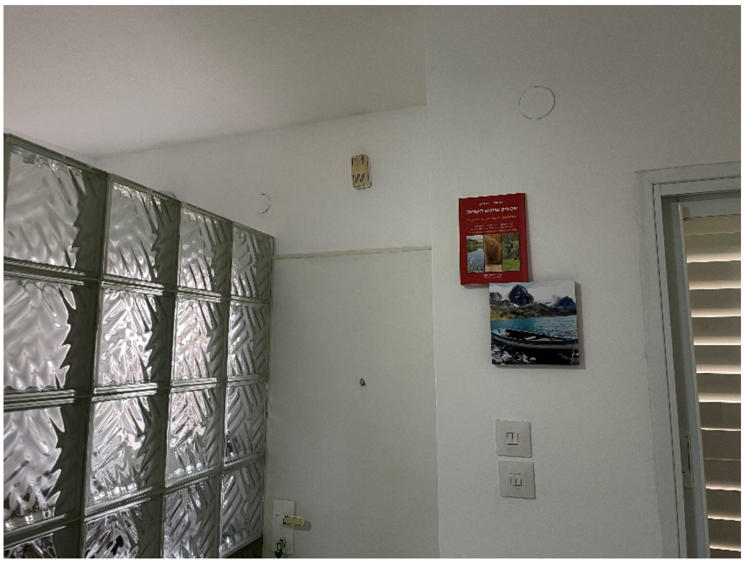

0
250
500
750
1000
1250
1500
1750
0
200
400
600
800
1000
1200
1400
יעקב סער   יגיל הנקין
שבילים ארוכים בישראל
לתרמילאים • לרוכבי אופניים • למשפחות
שביל ירושלים • המסע מים לים • שביל הבשורה
סובב באר שבע • שביל הגולן • שביל עמק המעיינות
כולל מפות טופוגרפיות של המרכז למיפוי ישראל
עדכון מידע: דני גספר
אשכול הוצאה לאור בע"מ

Original image



<u>Section 2 – Implement an image inside another image</u>

Following the scene we used with the book we changed:



Stages for creating the new image:

- Manually pick 4 corners of the book in the scene.
- Compute the homographic transformation between the reference image to the scene.
- Warping the reference image with the calculated transformation.
- Stitch the warp image with the scene image.

Following is one of the results images: