204219273 רועי פסטרנק
312545965 שני אמיר

Assignment 1:

Homography & Panorama

Part A: Homography computation

1. A system of equations of the form $A\underline{x} = \underline{b}$, for projective transformation:
Since we have matching points in 2 images, and we want to find the projective transformation between the two images- we want to find the Homography matrix corresponding for both images.
As we learned in class:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Therefore we get two linear equations per pair of patching feature points:

$$x' = \frac{a_1 x + a_2 y + a_3}{c_1 x + c_2 y + c_3}$$
$$y' = \frac{b_1 x + b_2 y + b_3}{c_1 x + c_2 y + c_3}$$

We have system of linear equations, $A\underline{x} = \underline{b}$, Where $\underline{x}$ is a vector of unknowns

$$\underline{x} = [a, b, c, d, e, f, g, h, i]^T$$

We need at least 8 equations, because $\underline{x}$ is up-to scale, but the more the better.
With more than 8 equations, the system is over constrained, thus solving using least squares:

$$min\left\|A\underline{x} - \underline{b}\right\|^2$$

For each pair of matching points, we get 2 equations.
For n pairs of matching points, we will receive the following system:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}_{2nx9} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}_9 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}_{2n}$$

Since h is only defined up-to scale, we will solve for unit vector $\hat{h}$.

The solution is $\hat{h}$ = eigenvector of $A^T A$ with the smallest eigenvalue.

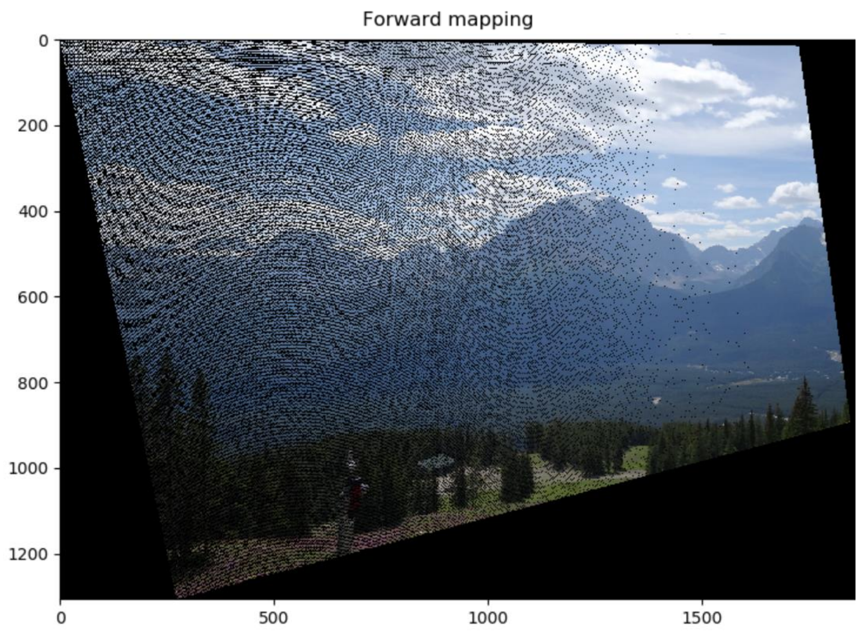
רועי פסטרנק 204219273
שני אמיר 312545965

Once we have found $\hat{h}$, we can find the conversion matrix H with re-ordering its shape.

2. Implemented in code
3. Result for matches_perfect.mat:

```
[[    1.43457214     0.21044323 -1277.18679001]
 [    0.01342652     1.34706123   -16.04558722]
 [    0.00037928     0.00005565     1.        ]]
```

4. The source image after a projective transformation, using the Forward Mapping transform:



5. The problems with Forward Mapping, is when a pixel from the source image, is mapped to the destination image, it is not guaranteed to fall on exact pixel. We will get a partial-pixel's value, and we wouldn't know which pixel matches this value.

   Another problem is, that we won't fill all the pixels in the destination image. There's a different scale that causes holes between the mapped pixels.

   In our image, we used forward mapping with bilinear interpolation to cover the black holes (pixels that had no value after the mapping).

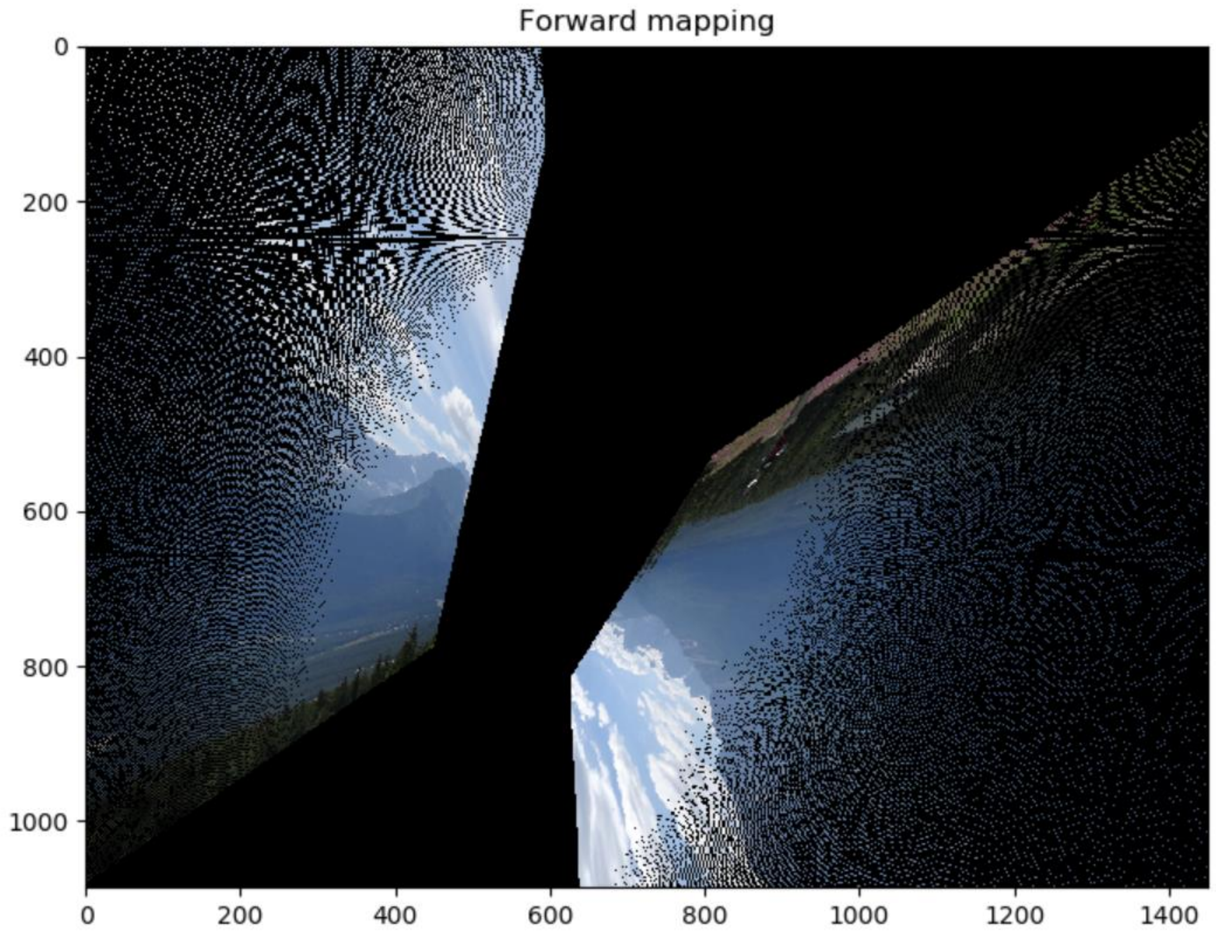
   We can see the image has black holes that represent the unmapped indexes, and places with high- density that represent pixels mapped from more than 1 source pixel.

רועי פסטרנק 204219273
שני אמיר 312545965

6. Result for matches.mat:

```
[[ -0.58601838  -0.13125975 625.47963816]
 [ -0.62585177  -0.48527678 817.14319931]
 [ -0.00094802  -0.0003852    1.        ]]
```

The source image after a projective transformation, using the Forward Mapping transform:



The outcome that has been received is completely different from the matches_perfect result.

The outliers changed the homography matrix and caused a false mapped image.

204219273 רועי פסטרנק
312545965 שני אמיר

Part B: Dealing with outliers

7. Implemented in code

8. Implemented in code

9. Given 30 matching points and that 80% of them are correct, the number of randomizations to guarantee 90% confidence is according to the following formula:

$$k = \frac{\log(1-p)}{\log(1-w^n)} \text{ } where\ p = 0.9, w = 0.8\ and\ n = 4\ hence: k = 4.37 \rightarrow 5\ iterations$$

Calculating the same for 99% confidence (p=0.99):

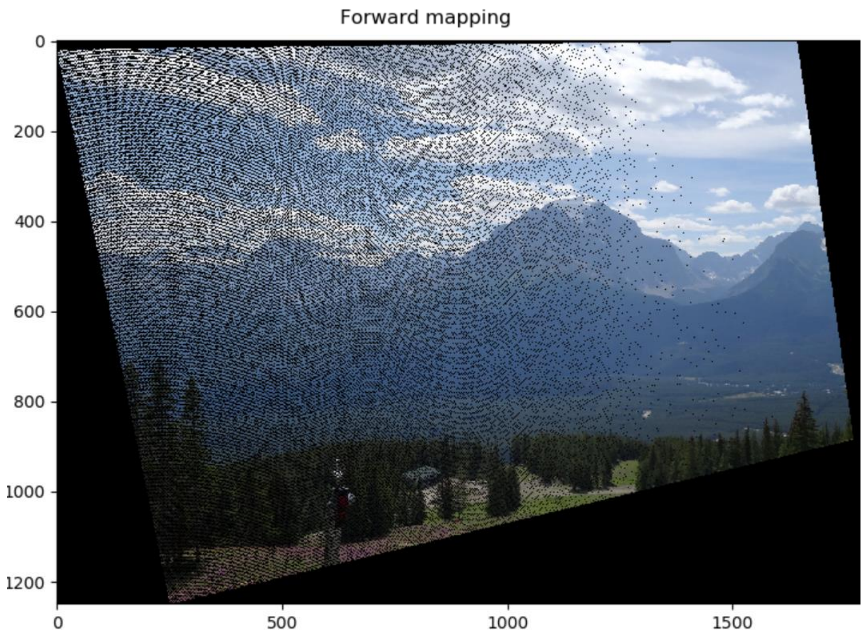
$$k = 8.74 \rightarrow 9\ iterations$$

And for covering all options for the given 30 points we need to preform: $k = (30\ choose\ 4) = 27405$ iterations.

10. Result for matches.mat using the :

```
[[   1.36787816    0.202984   -1217.63434816]
 [  -0.02907045    1.30238728    29.38765542]
 [   0.00031436    0.0000803      1.        ]]
```

The source image after a projective transformation, using the Forward Mapping transform:



204219273 רועי פסטרנק
312545965 שני אמיר

First lets' compare the coefficients obtained in each section using MSE distance:

$$\mathrm{H}_4 - naive\ computation\ using\ matches\_perfect.dat$$

$$\mathrm{H}_6 - naive\ computation\ using\ matches.dat$$

$$\mathrm{H}_{10} - computation\ using\ matches.dat\ and\ RANSAC$$

$$MSE(\mathrm{H}_6, \mathrm{H}_4) \cong 479372$$

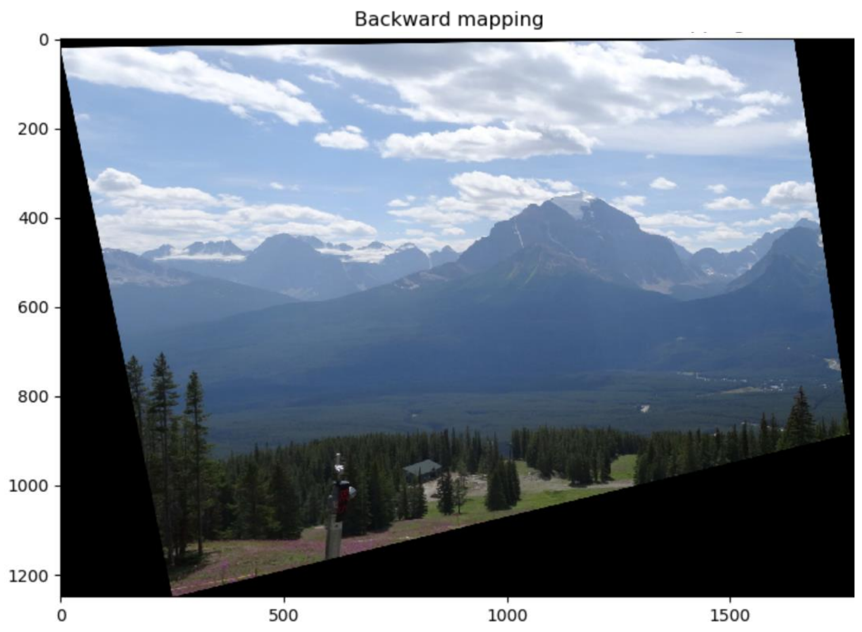
$$MSE(\mathrm{H}_{10}, \mathrm{H}_4) \cong 947$$

We can see that after using the RANSAC to remove the outliers from the matchs.dat we receive much better result than without the RANSAC. The MSC distance has been got smaller in three order of size. We used MSE since it measures the distance elementwise.

Furthermore, we can see that after applying the RANSAC algorithm we received pretty much the same image as in section 4 (up to some minor rotation delta). We still need to resolve the artifacts that we received in all the results until now.

Part C: Panorama creation

11. The source image after a projective transformation using backward mapping, according to the coefficients obtained in section 10:

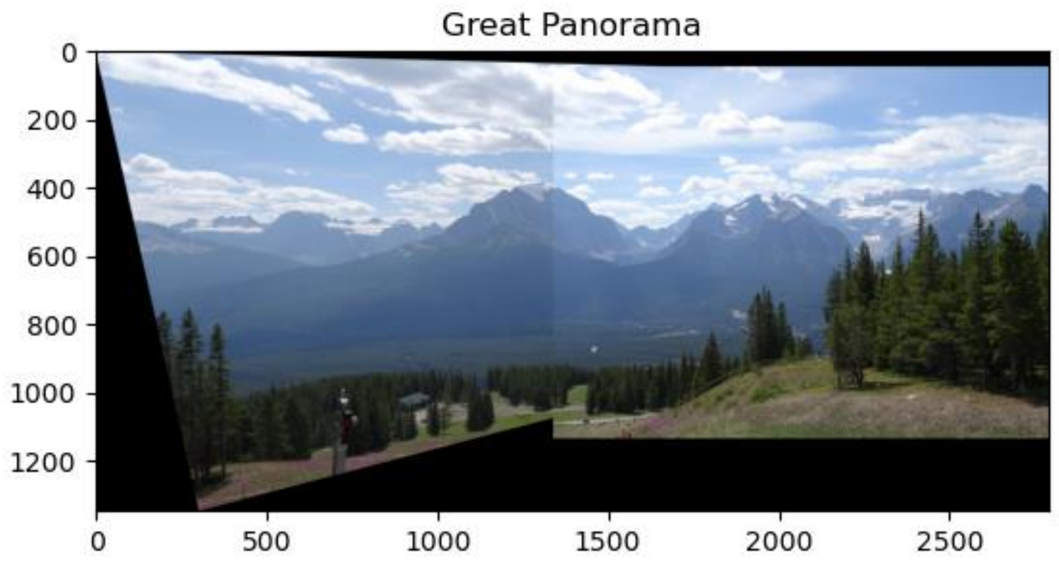


We can see that using the backward mapping resolved the "holes" artifacts we had when used the forward mapping – as expected.

רועי פסטרנק 204219273
שני אמיר 312545965

12. Implemented in code

13. Result of: *panorama(img_src, img_dst, match_p_src, match_p_dst, inliers_percent=0.8, max_err=25)* using the match.dat



רועי פסטרנק 204219273
שני אמיר 312545965

14.

Source image

Destination image

Panorama result: